AD

PAA-TRI-72

THE NUMERICAL SOLUTION OF TRANSIENT QUEUEING PROBLEMS

TECHNICAL REPORT

BY

STUART W. OLSON

SYSTEMS ANALYSIS DIVISION
PLANS AND ANALYSIS DIRECTORATE
HQ, US ARMY WEAPONS COMMAND
ROCK ISLAND, ILLINOIS

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Systems Analysis Division Plans and Analysis Directorate HQ, US Army Weapons Command	UNCLASSIFIED
	2b. GROUP

3 REPORT TITLE

THE NUMERICAL SOLUTION OF TRANSIENT QUEUEING PROBLEMS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Final Report

5 AUTHOR(S) (First name, middle initial, last name)

Stuart W. Olson

6 REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May 1972		34

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	PAA-TR1-72
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10 DISTRIBUTION STATEMENT

Distribution of this document is unlimited

11 SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
Master's Thesis	HQ, US Army Weapons Command Rock Island, Illinois 61201

13. ABSTRACT

This report explores methods for obtaining transient solutions to queueing problems which can be represented in the form of differential-difference equations. Six distinct methods, representing the most frequently-encountered in the open literature, are discussed as to their value in numerical work. The method of Runge-Kutta integration of these equations was found to be superior to the numerical evaluation of analytic solutions of a particular queueing model.

A generalized, Runge-Kutta programming package, written in FORTRAN IV for the IBM 360/65, is presented and described in detail for use on queueing problems. Generality is achieved by requiring the user to write a subroutine to evaluate his queueing equations when required by the programming package. Several sample problems are presented and solved to demonstrate the wide potential applicability of this method and associated computer program. The advantage of this method is that the usual rate parameters to describe state transitions may be any functions freely-chosen by the user.

The appendix which contains the FORTRAN program and instructions for its use may be separated from the report and used separately.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Queueing Theory						
Transient Solution Methods						
Numerical Methods						
Computers						
Programming						
FORTRAN						

THE NUMERICAL SOLUTION OF TRANSIENT QUEUEING PROBLEMS

By

Stuart W. Olson

Systems Analysis Division
Plans and Analysis Directorate
HQ, US Army Weapons Command
Rock Island, Illinois

May 1972

PAA-TR1-72

ABSTRACT

This report explores methods for obtaining transient solutions to queueing problems which can be represented in the form of differential-difference equations. Six distinct methods, representing the most frequently-encountered in the open literature, are discussed as to their value in numerical work. The method of Runge-Kutta integration of these equations was found to be superior to the numerical evaluation of analytic solutions of a particular queueing model.

A generalized, Runge-Kutta programming package, written in FORTRAN IV for the IBM 360/65, is presented and described in detail for use on queueing problems. Generality is achieved by requiring the user to write a subroutine to evaluate his queueing equations when required by the programming package. Several sample problems are presented and solved to demonstrate the wide potential applicability of this method and associated computer program. The advantage of this method is that the usual rate parameters to describe state transitions may be any functions freely-chosen by the user.

The appendix which contains the FORTRAN program and instructions for its use may be separated from the report and used separately.

ACKNOWLEDGEMENTS

This report is identical to the author's Master's thesis for the Department of Management and Industrial Engineering at the University of Iowa. The author is deeply indebted to the Army Weapons Command which provided support during the study program and particularly during the preparation of the thesis.

The author wishes to gratefully acknowledge his thesis supervisor Professor Liittschwager for his valuable stimulation and guidance during the preparation of this thesis and for his foresight in suggesting this topic for study.

The author also acknowledges the earlier stimulation and guidance of Mr. George Schlenker who originally suggested the approach to the problem.

The author greatly appreciates the services of Sandi Conley, who typed the manuscript.

TABLE OF CONTENTS

Page

LIST OF FIGURES v

LIST OF TABLES vi

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 REVIEW OF METHODS FOR

PARTICULAR TRANSIENT QUEUES 3

Generating Functions 8

Analytic Continuation 10

Power Series Expansions 11

Integral Equations 13

Monte-Carlo Simulation 17

Runge-Kutta Integration 18

CHAPTER 3 A COMPARISON OF THREE METHODS

FOR THE M/M/1 QUEUE 23

Analytic Bessel Function Method 23

Runge-Kutta Method 25

Monte-Carlo Simulation 32

CHAPTER 4 EXAMPLES OF SOLUTIONS TO OTHER

QUEUEING MODELS 34

Preemptive-Resume Queue 34

State-Dependent Queue 46

CHAPTER 5 SUMMARY 55

LIST OF REFERENCES 58

GENERAL REFERENCES 60

APPENDIX A LISTING OF MONTE-CARLO SIMULATION PROGRAM 62

APPENDIX B LISTING OF BESSEL FUNCTION PROGRAM 75

APPENDIX C RUNGE-KUTTA PROGRAM 88

Solution of the M/M/1 Queue 93

Subroutine INIT 94

PRIME Entry Point 97

NEWVAL Entry Point 98

WRAPUP Entry Point 99

Dimensioning the Problem 113

Program Output 116

Debugging Aids 117

Job Control Cards 118

LIST OF FIGURES

Figure		Page
1	CPU Time Versus Number in the System N for Bessel Function Method	27
2	CPU Time Versus Number in the System N for Runge-Kutta Method	28
3	Premptive-Resume M/M/1 λ=0.067 , μ=0.333 λ_p=0.00167 , μ_p=0.0333 Expected Number and Standard Deviation of Regular and Priority Systems	45
4	1-channel, 2-servers with ρ=1.5 , $0 \leq \tau < 8$ then ρ=0.5 , $\tau \geq 8$ for b=c=0	54

LIST OF TABLES

Table		Page
1	Comparative CPU Times	26
2	Summary of Benefits	30
3	Program Logic	92

CHAPTER 1

INTRODUCTION

This thesis explores problems associated with obtaining the transient solutions of queueing problems with varying rate parameters and various solution methods that have been employed. The work of Bhat (2) and Neuts (11) has shown that the lack of numerical work in the solution of transient queueing models has "handicapped practitioners to obtain meaningful results" (2). This thesis addresses that problem by presenting a complete, well-tested computer programming package for obtaining Runge-Kutta solutions to transient queueing problems. The thesis concludes that Runge-Kutta integration of the differential-difference equations of queueing problems is "best" in several respects, given the current state-of-the-art of available computational facilities.

Chapter two deals with a review of the computational aspects of several techniques contained in the literature. These include closed-form analytic solutions, approximations, simulation and Runge-Kutta integration. In Chapter three, an evaluation of three basic methods is made on the basis of computer programs which were written, tested and analyzed for each. This Chapter utilizes the single-channel, single-server queue as the principle comparative

model. Examples of solutions to other queueing models are then presented in Chapter four. The thesis is then summarized in Chapter five, followed by Appendicies A, B and C which present listings of computer programs used in Chapters three and four.

The programming package for Runge-Kutta solution of queueing problems is presented in detail in Appendix C. Included with it are complete instructions to guide users in its application to their problems and a detailed explanation of a sample problem.

CHAPTER 2

REVIEW OF METHODS FOR PARTICULAR TRANSIENT QUEUES

Queueing problems are frequently modeled in the form of a set of simultaneous, first-order, differential equations called the Chapman-Kolmorgorov (5) equations. These are

$$\dot{\underline{P}}(t) = A\underline{P}(t),$$

where $\dot{\underline{P}}(t)$ and $\underline{P}(t)$ are column vectors and $A = \{a_{ij}\}$ is a system transition matrix representing the time-rate of a transition from state i to state j. If $\underline{P}(t) = [P_0(t), P_1(t), \cdots, P_N(t)]$, then $P_n(t)$ represents the probability that the system is in state n at time t. Correspondingly, $\dot{P}_n(t)$ represents the time rate of change of $P_n(t)$. In queueing, the system is usually thought of as a set of channels and servers, related by a queue discipline which describes the flow of items in the system. Thus the notation $\dot{P}_n(t)$ describes the time fluctuation of the probability of seeing n items in the system for $0 \leq n < \infty$.

For the M/M/1 queue (15), the general equation for $n \geq 1$ in the system is

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda+\mu)P_n(t) + \mu P_{n+1}(t).$$

The resulting matrix form, limiting the maximum number in the system to N is

$$\begin{bmatrix} \dot{P}_0(t) \\ \dot{P}_1(t) \\ \dot{P}_2(t) \\ \cdots \\ \dot{P}_N(t) \end{bmatrix} = \begin{bmatrix} -\lambda & \mu & 0 & \cdots & 0 & 0 \\ \lambda & -(\lambda+\mu) & \mu & \cdots & 0 & 0 \\ 0 & \lambda & -(\lambda+\mu) & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & & -(\lambda+\mu) \end{bmatrix} \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \\ \cdots \\ P_N(t) \end{bmatrix} .$$

In the transient queueing problems considered by this thesis it is desired to admit rate parameters a_{ij} of any arbitrary form and functional dependence on either time t or number in the system n. Many analytic techniques require that n tend to infinity for a solution. In this thesis, the Runge-Kutta methods discussed will require n be definitely bounded at some finite value N. Further consideration regarding this point will be presented in Appendix C.

All too frequently, only steady-state results are considered, i.e., the solution of

$$\underline{0} = A\,\underline{P}$$

even when the system under study has frequent inherent changes in arrival or service rates. Numerous situations can be found where arrival rates, service rates and perhaps other system parameters are changing continually. One such situation is in a repair shop. Initially, the shop is empty at the start of the day. Arrivals occur at a rate greater than the service rate during the day. Steady state solutions to such systems are inadequate. What is needed is a method which will yield transient solutions to such problems with no more difficulty than that required for steady state solutions. That method is presented in this thesis.

The argument for seeking a standard technique for solving a wide class of transient queueing problems has a strong practical appeal. The technique can be studied on problems with known solutions in order to determine its properties under wide ranging conditions. The analyst can thus develop intuitive insights which are helpful when applying the technique to problems whose solutions are not known. In applying a standard solution technique, however, the analyst runs the risk of overlooking a possible analytic solution had he tried some other methods. In Chapter three the economics of attempting to solve the analytic closed form of a transient queueing problem versus using Runge-Kutta and Monte-Carlo techniques will be discussed further.

A search of the literature reveals that transient solutions to numerous queueing problems can be found.

However, in most instances these solutions appear in graphical form with no reference to the numerical techniques used to obtain them. In cases where the numerical techniques employed are revealed (12, 13) little discussion has dealt with the reason for selecting the particular technique from among the many available. Therefore, the literature concerning explicit techniques for transient queues is quite limited although at least eight techniques have appeared in the open literature. Six of the most useful are

1. Use of generating functions (z-transforms)
2. Analytic continuation of the differential equations
3. Series expansion of rate parameters
4. Transformation to integral equations
5. Monte-Carlo simulation
6. Direct integration of the differential equations

Two additional methods do not appear promising for study here. These are the method of numerical inversion of the Laplace transform and the method of eigenvectors. The method of numerical inversion does not offer the desired qualities of generality and numerical stabilty. The Laplace transform is difficult to obtain when rate parameters are functions of time. Also, in a thesis by Schlenker (16, p.14) the numerical method proposed by Bellman is shown to be frequently unstable and limited in accuracy.

The method of eigenvectors also does not seem promising. In a paper by Kaspar (8), the method is

discussed on a multiple, parallel channel system with no queues allowed. Kaspar claims that if the A matrix can be shown to have distinct eigenvalues, then a new system of linear differential equations with separated variables can be obtained. In our problem, however, the matrix A may be time-varying in such a way as to cause this method to fail. Therefore this method is of no further interest in this thesis.

Many of these methods have been reviewed by Leese and Boyd (10). They recommend an integral equation technique as the best method to solve the M/M/1 queueing problem based on computational time and storage considerations. This chapter draws upon their review and computational experience.

The six more useful methods are now discussed in greater detail. In the first four, the model to be used is the M/M/1 queue with equations written as

$$\dot{P}_0(t) = -\rho(t)P_0(t) + P_1(t) \quad , \text{ and for } n>1$$

$$\dot{P}_n(t) = \rho(t)P_{n-1}(t) - (\rho(t)+1)P_n(t) + P_{n+1}(t) \quad ,$$

where $\rho(t) = \lambda(t)/\mu$ and t is measured in units of μ.

Generating Functions

In this section, transient solutions obtained by application of generating functions are discussed. Using the equations for $P_n(t)$ as defined above, the following generating function

$$F(x,t) = \sum_{n=0}^{\infty} x^n P_n(t)$$

may be defined. Partial differentiation with respect to x and t and elimination of $P_n(t)$ terms yields the Fokker-Planck Diffusion equation

$$x(1-x)\frac{\partial^2 F}{\partial x \partial t} - (1-x)^2(1-\rho x)\frac{\partial F}{\partial x} + \frac{\partial F}{\partial t} + \rho(1-x)^2 F = 0.$$

Solutions to this equation with finite difference techniques are frequently difficult to obtain due to instabilities. Other techniques, such as the method of characteristics and the particle-in-cell method have proven unsuccessful in various degrees when $\rho(t)$ is nonlinear with respect to t.

This technique has only thus succeeded in transforming a difficult problem into an awkward problem for the numerical analyst. Computationally, partial differential equations also require large amounts of core storage, making them expensive to solve as well. Also, the method makes explicit use of the differential equations which means that

for each model, a new derivation for the partial differential equation is required.

The use of a generalized generating function of the form

$$F(x,t) = \sum_{n=0}^{\infty} f_n(x,t) P_n(t)$$

can be sometimes used to advantage when a particular choice of f_n will simplify the resulting partial differential equation. For example, if f_n is a polynomial in x with degree n and $P_n(t)$ is as above, $F(x,t)$ will satisfy

$$\frac{\partial F}{\partial t} = \rho(x-1)F$$

if x satisfies

$$\rho f_{n+1} + f_n + f_{n-1} = (\rho x+1) f_n \quad , \quad n \geq 1$$

with

$$f_0 = 1 \quad \text{and} \quad f_1 = x.$$

In this case the second order partial differential equation is thus simplified to a first order system with an auxiliary algebraic condition.

Another means of solving partial differential equations is by the method of integral equations. This method, sufficiently important in its own right, is discussed in a separate section of this chapter starting on page 13.

Analytic Continuation

A discussion is now given of the method of analytic continuation of the function $P_n(t)$ in the interval $(t,t+h)$ using Taylor's expansion.

Knowing $P_n(t)$ for some t allows one to obtain $\dot{P}_n(t)$ for all n. Repeated differentiation of the equations $P_n(t)$ yields

$$\frac{d^{(j)}}{dt^{(j)}}[P_n(t)] \quad .$$

Applying Taylor's formula, the expression at t+h becomes

$$P_n(t+h) = \sum_{j=0}^{\infty} \frac{h^j}{j!} \frac{d^{(j)}}{dt^{(j)}} [P_n(t)] \quad .$$

At least eight terms of the series are usually required for meaningful accuracy. We need only note that if ρ is a function of t, the chain rule of differentiation will make calculation of the derivatives rather tedious. Although programs for doing this (FORMAC, SNOBOL, LISP, etc.) are available, they are expensive and difficult to use.

The use of this technique requires an error analysis of the contribution of truncated terms of the series. Usually, an attempt is made to bound the error with an expression involving the number of terms before truncation, h, and the magnitude of higher derivatives of $P_n(t)$ (1).

Power Series Expansions

This technique uses the Bessel function solution which is apparently originally due to Clark (3, 15). This solution is given here in Chapter three on page 24. The Bessel functions and exponential terms of this solution are rearranged such that $P_n(t)$ is expressed as a power series in ρ of the form

$$P_{N,n}(t) = \sum_{m=0}^{\infty} \rho^m C_{N,n}^m(t)$$

where $$C_{N,n}^m(t) = T_m(t)\nabla^m S_{N+m-n}(t) - T_{m-n}(t)\nabla^{m-n-1}S_{N+m}(t)$$

$$+ T_{m-n-1}(t)\nabla^{m-n-2}S_{N+m}(t) \quad ,$$

$$T_j(t) = \frac{t^j}{j!} \qquad (j \geq 0), \quad T_j(t) = 0 \quad (j < 0) \quad ,$$

$$S_j(t) = e^{-t}T_j(t), \quad \text{and}$$

$$-\nabla^{-1}S_{j-1}(t) = \sum_{u=j}^{\infty} S_u(t).$$

This method works well if ρ can be approximated by constant values over short, equally-spaced intervals, since the C's need only be computed once and stored for all N,m,n and t. However, a large amount of computing time and high-speed storage are required. Also, the method is not exact and requires a great deal of effort to establish the expressions for the C's if the form of the equation changes. The problem of numerical instability in evaluating the terms in the infinite sums must be constantly considered by the analyst, since the C's can exceed unity.

To avoid this problem, the terms in the power series expansion can be written differently leaving the Poisson part unaltered. Thus

$$P_{N,n}(t) = \sum_{m=0}^{\infty} \left[\frac{(\rho t)^m e^{-\rho t}}{m!}\right] F^m_{N,n}(t)$$

where

$$F^m_{N,n}(t) = S_{m+N-n}(t) + \frac{T_{m-n}(t) - T_{m-n-1}(t)}{T_m(t)} \sum_{u=m+N+1}^{\infty} S_u(t).$$

In this form, each of the terms [.] and F represent probabilities and are thus less than unity. Although the possible computational difficulties are largely avoided with this new form, obtaining a new set of equations for each different queueing problem encountered would be tedious.

Integral Equations

In this section, techniques will be discussed for solving partial differential equations such as the form of the Fokker-Planck equation developed in the section under Generating Functions. A solution to the Fokker-Planck partial differential equation may be formulated in terms of a Volterra-type integral equation; however it is not suitable for numerical work.

One formulation, depending upon a change of variable, has been found quite suitable for solution by computer, and in fact, is probably the most efficient in terms of speed and storage required. This is the method of Wragg as implemented by Leese and Boyd (10). The change of variable is

$$q_n(t) = \sum_{j=n}^{\infty} P_j(t) \ .$$

Then by substitution into the equations for $P_n(t)$

$$\dot{q}_n(t) = \rho q_{n-1}(t) - (1+\rho) q_n(t) + q_{n+1}(t) \ , \ n \geq 1$$

with $q_0(t) = 1$ and $q_n(0) = 0 \ , \ n \geq 1 \ .$

By defining

$$u = \int_x^t \rho(y)dy \ , \ v = \left[\frac{u}{t-x} \right]^{1/2} \text{ and } \omega = 2\, u(t-x)^{1/2}$$

and writing in original $q_n(t)$ form, we obtain

$$q_n(t) = \int_0^t [I_{n-1}(\omega)\rho(x)v^{n-1} - I_{n+1}(\omega)v^{n+1}]\exp(-u-t+x)G(x)dx.$$

In this expression $I_n(\omega)$ is a Bessel function of the First Kind and G(x) satisfies

$$1 = G(t) + \int_0^t K(t,x)G(x)dx ,$$

where

$$K(t,x) = [\frac{\rho(x)}{v} - v] \quad I_1(\omega)\exp(-u-t+x) .$$

This technique still involves solution of Bessel functions, complicated by the fact that solution for G(t) requires iteration. The above integral equation is far removed from the original model in terms of analytical transformations. To arrive at it, the form and assumptions of the original queueing problem were heavily used. Thus, altho this is an impressive method of solving for the transient conditions in the M/M/1 model, it can hardly be relied upon as a technique for a broad class of problems.

One other technique using partial differential equations for solution is useful in large-scale queueing systems. This method is due to Newell (12) who applied it in highway traffic problems.

Define $F(x,t) = P[X(t) < x]$ where $X(t)$ is the queue length at time t, then the Fokker-Planck equation may be used as a second-order approximation of $\frac{\partial F(x,t)}{\partial t}$ where

$$\frac{\partial F}{\partial t} = a(t) \frac{\partial F}{\partial x} + \frac{b(t)}{2} \frac{\partial^2 F}{\partial x^2} .$$

This is valid for large X, a large number of arrivals and departures in time interval τ with only a small change of X during τ and further, that the number of arrivals and departures during τ is normally distributed. These conditions are often not unduly restrictive for large scale systems.

Let $E[A(t)]$ be the number of arrivals until t and $E[D(t)]$ be the number of departures until t.

Then

$$\tau a(t) = E[A(t) - D(t)] - E[A(t+\tau) - D(t+\tau)]$$

$$= E[D(t+\tau) - D(t) + A(t+\tau) - A(t)]$$

and similarly

$$\tau b(t) = VAR[D(t+\tau) - D(t) + A(t+\tau) - A(t)].$$

By defining $t^* = \frac{t}{T_0}$ and $x^* = \frac{x}{L_0}$ then

$$\frac{\partial F}{\partial t^*}(x^*,t^*) = \frac{Ta(t^*)}{L}\frac{\partial F}{\partial x^*}(x^*,t^*) + \frac{Tb(t^*)}{2L^2}\frac{\partial^2 F(x^*,t^*)}{\partial x^{*2}}$$

Now if $T = T_0$ and $L = L_0$ at $t=0$, then

$$\frac{\partial F}{\partial t^*} = \frac{T_0}{L_0} a_0 \frac{\partial F}{\partial x^*} + \frac{T_0 b_0}{2L_0^2}\frac{\partial^2 F}{\partial x^{*2}} .$$

Choose b_0 such that $\frac{T_0 b_0}{2L_0^2} = \frac{1}{2}$ and

let $a(t) = -\alpha t$ so that

$$\frac{\partial F}{\partial t^*} = -\alpha\, t^* \frac{\partial F}{\partial x^*} + \frac{1}{2}\frac{\partial^2 F}{\partial x^{*2}} .$$

This equation is solved using a simple finite difference analogue similar to a random walk on a lattice. By letting

$$X_{j+1} = \begin{cases} X_j+1 & \text{with probability } p \\ \max(X_j-1,0) & \text{with probability } (1-p) \end{cases}$$

where

$$p_j = \frac{1}{2}(1+\alpha_j)$$

and with $F_j(k) = P\{X_j \leq k\}$, then

$$F_{j+1}(k) = p_j F_j(k-1) + (1-p_j)F_j(k+1) , \quad k \geq 1$$

and $F_{j+1}(0) = (1-p_j)F_j(1)$, $\alpha_j << 1$,

which can be applied iteratively to solve

$$\frac{\partial F_j(k)}{\partial j} = (1-\alpha_j)\frac{\partial F_j(k)}{\partial k} + \frac{1}{2}\frac{\partial^2 F_j(k)}{\partial k^2} .$$

Monte-Carlo Simulation

Monte-Carlo simulation is a powerful tool for solving queueing problems in the sense that it can presumably be applied to any model. In cases where only a narrative description of the problem exists or where differential equations can not be derived describing the model, Monte-Carlo is perhaps the only quantitative approach. The relevant question then is how efficient is Monte-Carlo as compared to numerical techniques? Modern computational equipment, simulation languages, variance-reduction techniques and refined cost accounting of computer programs have made Monte-Carlo simulation an attractive alternative for an analyst who may not wish to spend a great deal of his time attempting to derive a set of differential equations to describe his system. Monte-Carlo simulation of a simple model is used when it is contemplated that the model will be enriched with complexities that will make it analytically intractable at a later time.

Monte-Carlo simulation is a reasonable way of estimating the steady-state solution of a system. Estimating transient behavior, however, is much more difficult and costly if it is desired to estimate P(t) at many discrete values of t. Since, in reality, analysts are generally interested in obtaining quantities such as the waiting time, number in the system, and the number of units processed, P(t) is not required and the desired quantities may be observed directly and economically.

A simulation of a M/M/1 queue was written in SIMSCRIPT-II (.) for this thesis in order to form a basis for comparison with other methods. SIMSCRIPT-II is probably one of the most suitable and efficient languages available for simulation involving transient queueing behavior. The simulation and results are given in the next chapter.

Runge-Kutta Integration

Direct solution of the differential equations by Runge-Kutta methods is not only simple and straightforward, but well understood, flexible and easy to apply to new problems. The numerical analysis and behavior of Runge-Kutta are well-known. The method is the most stable of the methods commonly used in differential equations (Euler, Milne, Adams Bashforth, etc.), it is easy to use and produces error of order h^4, where h is the step size. This author has

written a subroutine which will integrate simultaneous differential equations of arbitrary size and order and has established several years of experience with it on wide classes of problems encountered in differential equations. The Chapman-Kolmogorov equations are a particularly "worry-free" class of problems solved by the Runge-Kutta method. Therefore, it seems reasonable to attempt the development of a general computer program to solve these equations. To a large extent, this has been accomplished.

The author's program consists of unit modules with multiple entry points. The user writes a subroutine named INIT with only the minimal amount of FORTRAN coding necessary to describe his equations and problem peculiarities. The first entry point is used for initialization. The second entry point is used to provide values for $\dot{P}_n(t)$, given $P_n(t)$ and t. The third entry point is used after each step in the solution to allow for computation of auxiliary quantities of interest such as queue statistics. The fourth entry point is used at the completion of the solution to re-initialize for another problem or to go on to some other task. These entry points will be discussed in greater detail in Appendix C. The programming of subroutine entry points is discussed in (6) in detail.

The Runge-Kutta subroutine, as described below, has been adapted following the form given by Nielsen in (14, p.

178). In this form, which is one of many alternative forms in the literature, it is possible to solve a set of n-th order simultaneous differential equations without explicitly redefining the higher order derivatives to obtain the usual first order system.

The Runge-Kutta subroutine performs four steps which involve evaluation of the differential equations to move the solution for $P_n(t)$ from t to $t+\Delta t$. Initially set

$$\Delta P_i = \dot{P}_i(t) \quad i=1,N.$$

Then the four Runge-Kutta steps appear as below.

R-K

Step No. Operations Performed

k=1 $\quad t_1 = t + \frac{\Delta t}{2} \qquad P_{i1} = P_i + \dot{P}_i \frac{\Delta t}{2}$; i=1,N

$\dot{P}_{i1} = \dot{P}_i(t_1)$; i=1,N $\qquad \Delta P_i \leftarrow \Delta P_i + 2\dot{P}_{i1}$; i=1,N

k=2 $\quad t_2 = t + \frac{\Delta t}{2} \qquad P_{i2} = P_i + \dot{P}_{i1} \frac{\Delta t}{2}$; i=1,N

$\dot{P}_{i2} = \dot{P}_i(t_2)$; i=1,N $\qquad \Delta P_i \leftarrow \Delta P_i + 2\dot{P}_{i2}$; i=1,N

k=3 $\quad t_3 = t + \Delta t \qquad P_{i3} = P_i + \dot{P}_{i2} \Delta t$; i=1,N

$$\dot{P}_{i3} = \dot{P}_i(t_3) \; ; \; i=1,N \qquad \Delta P_i \leftarrow \frac{\Delta t}{6}(\Delta P_i + \dot{P}_{i3}) \; ; \; i=1,N$$

k=4 $\quad t_4 = t + \Delta t \qquad P_{i4} = P_i + \Delta P_i \; ; \; i=1,N$

$$\dot{P}_{i4} = \dot{P}_i(t_4) \; ; \; i=1,N$$

$$t = t_4 \qquad P = P_{i4} \; ; i=1,N \qquad \dot{P}_i = \dot{P}_{i4} \; ; i=1,N$$

It should be noted that only three working storage vectors of length N need be provided for the P_i, P_{ik} and $\dot{P}_{ik}$ values. These are provided by the user's subroutine and may be used for other storage requirements when not in use by the four-step Runge-Kutta computations.

Rules for choosing Δt have been developed to allow it to change as the solution progresses, depending on the magnitude of some measure of truncation error (1). Generally, these rules are heuristic or highly problem-dependent. This author recommends a constant Δt initially and then re-running the problem at a few selected values for Δt, which should give the user a good feel for the trade-off between computational time (cost) versus quadrature error. Since the P_i values must sum to unity for any and all t, the difference between this sum and unity makes an excellent measure of accumulated error at each step of the solution.

Further work on this problem should consider the optimality of the weights of the $\dot{P}_{ik}$ in computing the

ΔP_i above. Currently,

$$\Delta P_i = \Delta t(a\dot{P}_i + b\dot{P}_{i_1} + c\dot{P}_{i_2} + d\dot{P}_{i_3})/w$$

where $a=1, b=2, c=2, d=1$ and $w=a+b+c+d$. These values are optimal if the function being integrated is a parabola on the interval from t to $t+\Delta t$. If one were to replace this assumption by the fact that the P_i must always sum to unity, it might be possible to derive a new set of weights more specifically suited to this kind of problem.

The next chapter will present a detailed comparison of a method representative of those discussed in this chapter, the Runge-Kutta method and Monte-Carlo simulation.

CHAPTER 3

A COMPARISON OF THREE METHODS FOR THE M/M/1 QUEUE

In this chapter a comparison is made between a representative analytic solution, the Runge-Kutta solution of a single-channel, single-server queue, and the Monte-Carlo method. The analytic solution for this model is presented in Saaty (15) and is considered representative in complexity and computational difficulty of all the analytical methods discussed in the previous chapter. The comparison of methods will consist of computing $\dot{P}(t)$ from t=0 until steady state is achieved, i.e., t=t* where $\dot{P}(t^*) \simeq 0$.

Analytic Bessel Function Method

The time-dependent solution of the single-channel, single-server queue is derived (apparently by Clarke (3)) in terms of modified Bessel functions of the First Kind.

This is given by Saaty as

$$P_n(t) = \exp[-(\lambda+\mu)t]\,[(\lambda/\mu)^{\frac{n-i}{2}} I_{n-i}(2\sqrt{\lambda\mu}\,t) + (1-\lambda/\mu)(\lambda/\mu)^n \sum_{k=n+i+2}^{\infty} (\lambda/\mu)^{\frac{-k}{2}} I_k(2\sqrt{\lambda\mu}\,t) + (\lambda/\mu)^{\frac{n-i+1}{2}} I_{n+i+1}(2\sqrt{\lambda\mu}\,t)],$$

where i is the initial number of items in the system. A computer program was written in FORTRAN using the Bessel function subroutine BESI from the IBM Scientific Subroutine Package (SSP) (7). The listing of this program with sample output is included in Appendix B. The size of the load module was 29,856 bytes. Time t was varied from 0 to 20 in steps of 0.1 and n varied from 0 to 20 with $i=0$. The parameters λ and μ were chosen as 0.45 and 0.5, respectively. The equation for $P_n(t)$ above was solved for 201 values of t and 21 values of n for a total of 4221 times. The amount of central processor unit (CPU) time for the IBM 360/65 was 98.64 seconds. Approximately four hours were required to write the program and several trials were needed to debug it. The infinite series term of the equation was truncated when either $k=n+50$ or the k-th term was less than 10^{-6}. This rule produced values of $P_n(t)$ which were no more than about 1% in error.

Runge-Kutta Method

The single-channel, single-server equations were integrated using the Runge-Kutta method. For completeness they are listed here again as

$$\dot{P}_0(t) = -\lambda P_0(t) + \mu P_1(t)$$

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda+\mu) P_n(t) + \mu P_{n+1}(t) \quad \text{for } 1 \leq n \leq N$$

$$\dot{P}_N(t) = \lambda P_{N-1}(t) - (\lambda+\mu) P_N(t) \quad \text{with} \quad P_0(0) = 1.$$

The same values for t, λ and μ were used with $N=20$. The time step Δt was chosen as 0.1. Since the Runge-Kutta method requires four evaluations of the equations per step, the above equacions were evaluated a total of $201 \times 4 = 804$ times for each value of n. This is $804 \times 21 = 16{,}884$ equations evaluated for the run. The Runge-Kutta method requires storage of the entire $\underline{P}(t)$ and $\dot{\underline{P}}(t)$ vectors plus three more working vectors of the same length. The size of the load module for this program was 25,960 bytes. The amount of CPU time for the solution was 7.42 seconds. The solution error grew to a maximum of 0.005% with this step size. Only the differential equations and initial conditions had to be programmed as a subroutine named INIT since the main program is completely general. This consumed about 1/2 hour and did not need to be debugged.

The following table illustrates how CPU time varies as a function of the maximum number in the system N for the Bessel function solution and for Runge-Kutta solution. Times are in CPU seconds.

Table 1

Comparative CPU Times

N	Bessel Functions	Runge-Kutta
10	59.0	5.0
20	100.9	7.4
30	149.9	9.6
40	205.2	11.3
50	290.6	12.9
75	677.9*	16.4
100	1428.5*	21.8

The asterisk (*) notation above indicates these figures are linear extrapolations based upon runs over a shorter range of t, since there is a uniform correspondence between CPU time for the run and t.

When plotted, the times for the Bessel function solution vary as a power function with N (Figure 1). The Runge-Kutta solutions vary linearly with N (Figure 2).

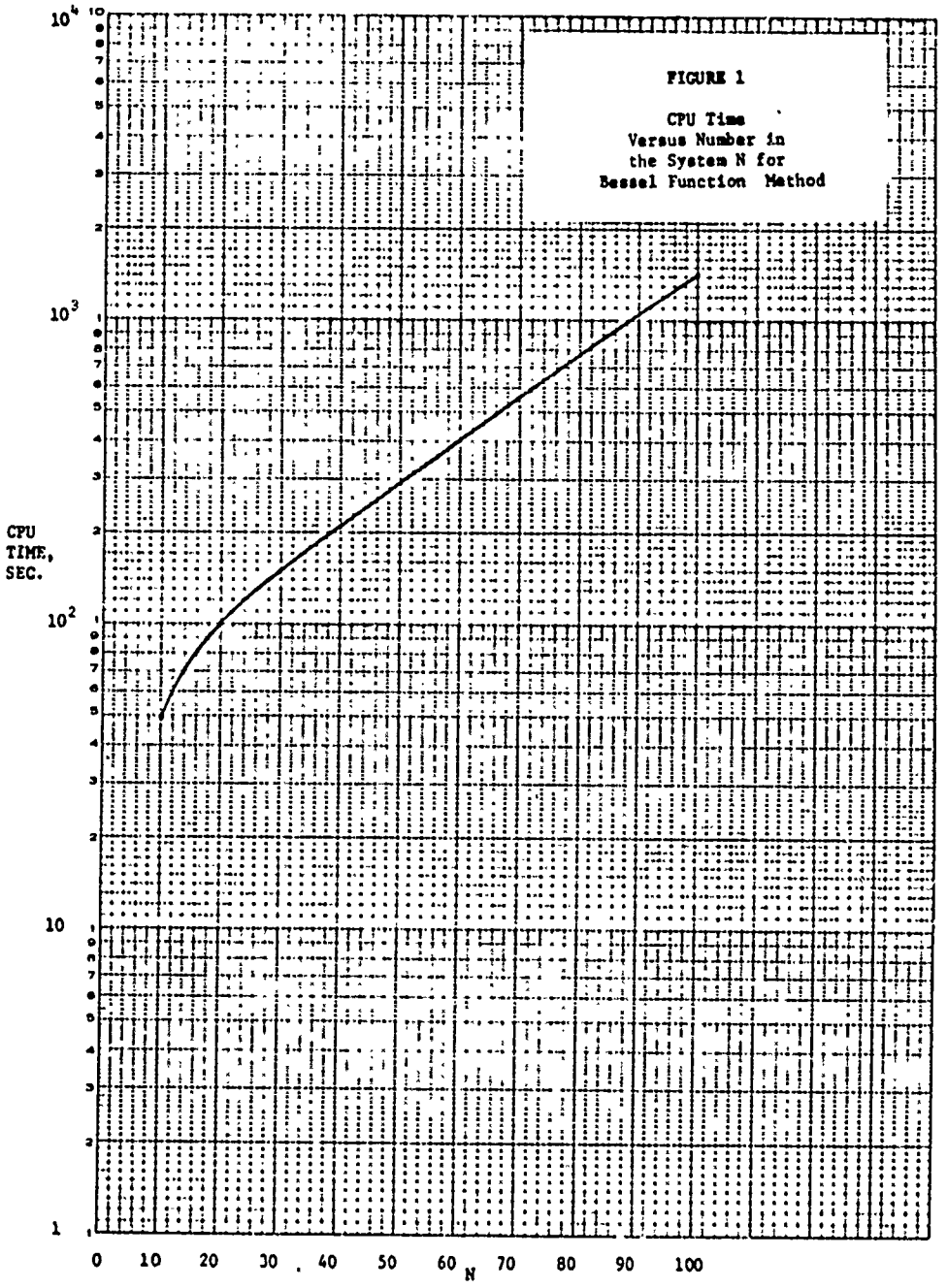


FIGURE 1
CPU Time
Versus Number in
the System N for
Bessel Function Method
CPU
TIME,
SEC.
10^4
10^3
10^2
10
1
0
10
20
30
40
50
60
70
80
90
100
N

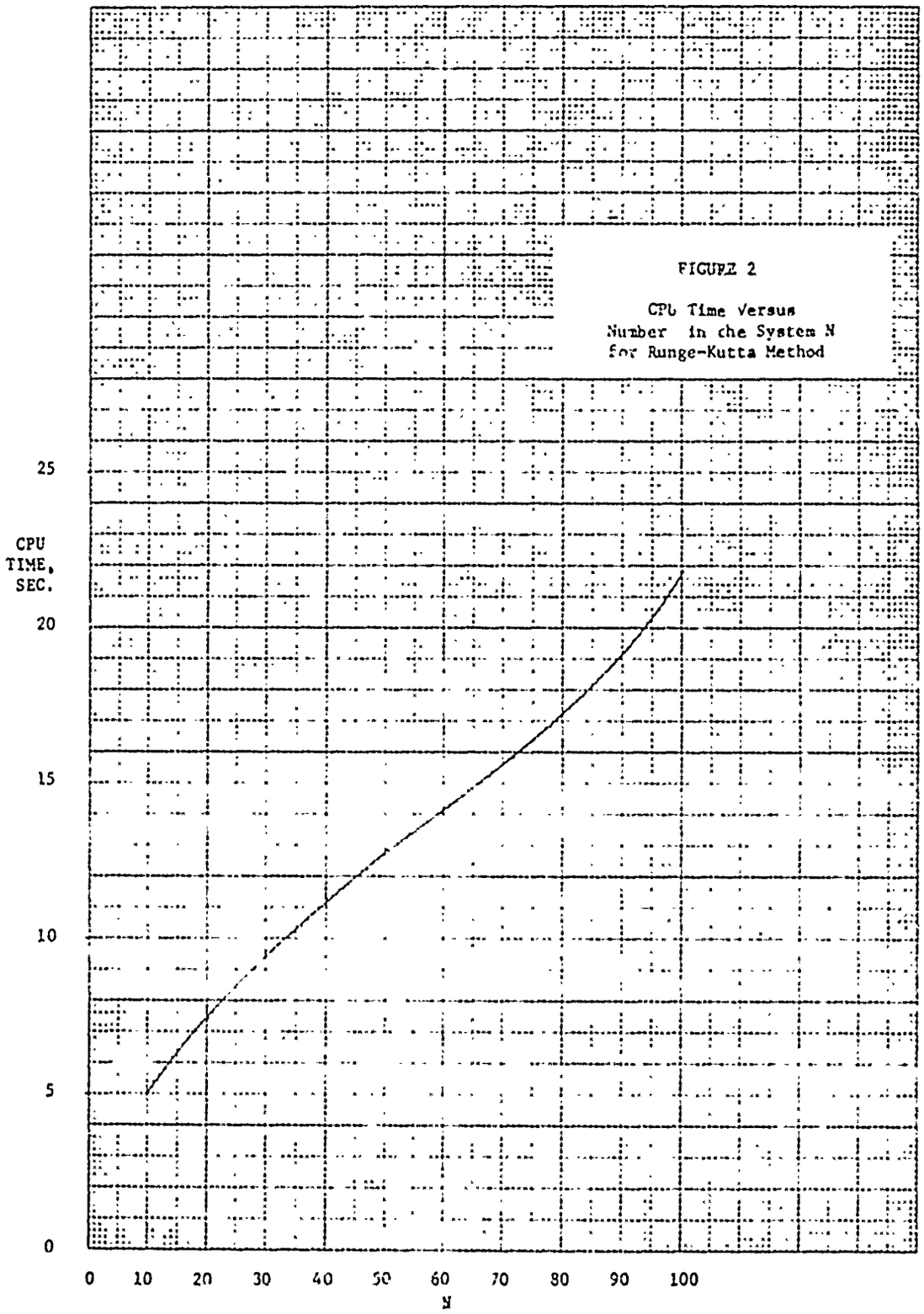


FIGURE 2

CPU Time Versus
Number in the System N
for Runge-Kutta Method

Estimates of required CPU time T may be obtained from the empirical relations

$$T = 46 \exp(0.0355N)$$

for Bessel functions, and

$$T = 4.2 + 0.172N$$

for Runge-Kutta. These were derived from the plots of the above CPU times. The cost of the additional core storage required to use the Runge-Kutta method beyond the break-even point of N=500 is insignificant in comparison to the increased CPU time cost required by the solution by Bessel functions.

The FORTRAN subroutine INIT for the M/M/1 queue discussed in this chapter is presented in Appendix C along with sample output used in the comparison.

The following table is a summary of benefits realized by using the Runge-Kutta method over the analytic solution involving Bessel functions. Each benefit more or less applies to every analytical technique discussed in the previous chapter.

Table 2

Summary of Benefits

Runge-Kutta	Bessel Functions
analytical difficulty	
No further analysis required beyond derivation of equations.	Considerable analysis to get analytic solution to equations. Analytic expression may not exist - may need approximations.
programming	
Easy to program for computer solution - little debugging required.	Sophisticated programming expertise required to prevent overflows, loss-of-significance, etc. Difficult to debug.
numerical stability	
Solution can be trusted, method behaves in a predictable way, precision can be arbitrarily chosen by user.	Numerical instabilities leave trustworthiness of solution in doubt, very little flexibility in choosing desired precision.

cost of accuracy	
Cost of achieving a given level of precision is related linearly to step size over wide range of values.	Cost of achieving a given level of precision usually is related non-linearly in a power function over a narrow range of precision.
analysts' time	
Time to obtain solution, given a model, is short.	Time to obtain solution, given a model, may be very long.
clarity of methods	
Method and solution easily understood by others.	Method chosen and implementation difficult for others to understand.
cost of solutions	
Cost of solution increases linearly with N , the number of equations solved.	Cost of solution increases as a power function of N.
size of programs	
Load module of equal size occurs at approxi-	Load module size does not change with N; greater for

mately N=500. N < 500 than the Runge-Kutta load module.

Monte-Carlo Simulation

A simulation of the M/M/1 model was written in SIMSCRIPT-II (9). The listing and sample output are contained in Appendix A. For a sample size of 1000 replications, consisting of 10 runs of 100 replications each, the 95% confidence interval on the expected number in the system n was obtained as

$$1.66 \leq n \leq 2.32.$$

Although this is quite a large variability, it is acceptable to achieve this level in complex simulations. To make a comparison with the other two methods on the basis of computer resources, the load module size was 48,000 bytes and the total CPU time was 28 seconds. This compares favorably with Runge-Kutta and shows the efficiency of the SIMSCRIPT-II system. The CPU time, of course, would depend linearly on the number of replications. The time to program and debug the simulation was slightly less than that required for the Bessel function program.

Although not strictly a numerical technique, the simulation was included to demonstrate its relationship with the previous two methods because simulation is usually

regarded as an expensive and inefficient method to solve such problems.

CHAPTER 4

EXAMPLES OF SOLUTIONS TO OTHER QUEUEING MODELS

In this chapter two other applications of the Runge-Kutta method will be presented. The purpose is to show how simple and straightforward the preparation for solution of different problems can be when using the Runge-Kutta control program in Appendix C. The first is the solution of the premptive-resume priority queue and the second is the solution of the state-dependent queue. These were chosen as further examples of solution by the Runge-Kutta integration because they are representative of commonly-used models.

Premptive-Resume Queue

The premptive-resume queueing model is a slight variation of the M/M/1. In it, the possibility of a different kind of priority arrival is allowed in the system with a different arrival rate than the normal arrivals. When a priority arrival occurs, service is suspended if service is being performed on a normal service item, and it is set aside so that service may begin on the priority arrival. Service proceeds at a different rate for the priority item than for normal items. The server continues

to serve from the priority queue until empty, at which time the normal item that had been set aside will return to service and continue where it left off.

This is an excellent model for describing how a M/M/1 model would behave if the server were subject to breakdown and repairs were made to restore it to service.

In the model that follows λ and μ are the arrival and departure rates, respectively, for the normal system and λ_p and μ_p are parameters for the priority system. The equations are written for a finite number M in the priority system and N for the normal system. Moreover the subscript i stands for the number in the priority system and j stands for the number in the normal system.

$$\dot{P}_{i,j}(t) = -(\lambda+\lambda_p+\mu_p)P_{i,j}(t)+\lambda_p P_{i-1,j}(t) + \mu_p P_{i+1,j}(t) + \lambda P_{i,j-1}(t)$$

$$\dot{P}_{0,0}(t) = -(\lambda+\lambda_p)P_{0,0}(t) + \mu_p P_{1,0}(t) + \mu P_{0,1}(t)$$

$$\dot{P}_{0,j}(t) = -(\lambda+\lambda_p+\mu)P_{0,j}(t) + \lambda P_{0,j-1}(t) + \mu P_{0,j+1}(t) + \mu_p P_{1,j}(t)$$

$$\dot{P}_{i,0}(t) = -(\lambda+\lambda_p+\mu_p)P_{i,0}(t) + \lambda_p P_{i-1,0}(t) + \mu_p P_{i+1,0}(t)$$

$$\dot{P}_{M,N}(t) = -(\lambda+\lambda_p+\mu_p)P_{M,N}(t) + \lambda_p P_{M-1,N}(t) + \lambda P_{M,N-1}(t)$$

$$\dot{P}_{0,N}(t) = -(\lambda+\lambda_p+\mu)P_{0,N}(t) + \lambda P_{0,N-1}(t) + \mu_p P_{1,N}(t)$$

$$\dot{P}_{M,0}(t) = -(\lambda+\lambda_p+\mu_p)P_{M,0}(t) + \lambda_p P_{M-1,0}(t)$$

$$\dot{P}_{M,j}(t) = -(\lambda+\lambda_p+\mu_p)P_{M,j}(t) + \lambda_p P_{M-1,j}(t) + \lambda P_{M,j-1}(t)$$

Subroutine INIT for use with the program in Appendix C and sample output follow, along with a plot of some auxiliary queue statistics in Figure 3.

```
FORTRAN IV G LEVEL  2                    INIT                    DATE = 72103          18/32/31          PAGE 0001
0001                  SUBROUTINE INIT
                      C
                      C***  INITIALIZATION ENTRY POINT FOR PREMPTIVE-RESUME QUEUEING
                      C***  EXAMPLE.
                      C***  THE P-ARRAY IS DIMENSIONED FOR 2-STATES IN PRIORITY SYSTEM,
                      C***  FOR 21-STATES IN NORMAL SYSTEM AND IS OF ORDER 1. THEREFORE THE
                      C***  WORKING STORAGE ARRAY MUST BE 1.5 TIMES THE DIMENSION OF P -- OR
                      C***  2*21*21*1.5=1323.
                      C
0002                        COMMON NAME(20),TC,TMAX,DT,NPRNT,NEQ,P(2,21,21),W(126)
0003                        REAL LAM,LAMP,MU,MUP
0004                        READ (5,1,END=20) NAME,LAM,LAMP,MU,MUP,TMAX,DT
0005                    1 FORMAT (20A4/6F10.0)
0006                        PRINT 2,NAME,LAM,LAMP,MU,MUP
0007                    2 FORMAT (1H1,20A4//1HO 9X6HLAM=0A 7X6HLAM=0A-P 13X2HMU 11X4HMU-P/
                       1 2X 4F15.6)
                      C
                      C***  INITIALIZE THE REQUIRED VARIABLES IN COMMON, THE P-ARRAY AND
                      C***  ANY AUXILIARY PARAMETERS AND VARIABLES AS REQUIRED BY THE
                      C***  PARTICULAR PROBLEM.
                      C
0008                        NPRNT=0
0009                        NPRNT=5
0010                        M=21
0011                        N=2
0012                        NEQ=M*N
0013                        MM1=M-1
0014                        NM1=N-1
0015                        CEL=LAM+LAMP
0016                        CELMP=CEL+MUP
0017                        CELM=CEL+MU
0018                        LIM=3
0019                        IMPRT=0
0020                        DO 14 I=1,M
0021                        DO 14 J=1,N
0022                    14  P(I,J,1)=0.0
0023                        P(1,1,1)=1.0
0024                        ESYS=0.0
0025                        VSYS=0.0
0026                        AVQ=0.0
0027                        VQ=0.0
0028                        ESYSP=0.0
0029                        VSYSP=0.0
0030                        AVQP=0.0
0031                        VQP=0.0
0032                        PSUM=0.0
0033                        PRINT 3
0034                        PRINT 4,TC,ESYS,VSYS,AVQ,VQ,ESYSP,VSYSP,AVQP,VQP,PSUM
```

```
0035            3 FORMAT (1H1 9X1HT 4X6HE(SYS) 3X7HSTD DEV 6X4HAV Q 3X7HSTD DEV
                 1 3X7HE(SYSP) 3X7HSTD DEV 5X5HAV OP 3X7HSTD DEV 3X7H% ERROR)
0036            4 FORMAT (1H F10.2,8F10.5,F10.3)
0037              RETURN
                C
                C***  THE SECOND ENTRY USED TO COMPUTE THE DIFF.-DIFF. EQUATIONS
                C
0038              ENTRY PRIME(T,PT,K)
0039              REAL PT(2,21,2)
                C
                C***  DURING THE SOLUTION, IN THIS ENTRY POINT SECTION, THE PT-ARRAY
                C***  IS USED RATHER THAN THE P-ARRAY BECAUSE WE ARE NOW DEALING
                C***  WITH DIFFERENT STORAGE LOCATIONS -- IN THE WORKING STORAGE AREA.
                C
0040              PT(1,1,2)=RHS(MUP*PT(2,1,1)+MU*PT(1,2,1)-CLLP*PT(1,1,1))
0041              IF (NMO.LE.1 .OR. NMO.LE.1) GO TO 15
0042              DO 5 I=2,NMO
0043                DO 5 J=2,NMO
0044            5 PT(I,J,2)=RHS(LAMP*PT(I-1,J,1)-CLLPMP*PT(I,J,1)+MUP*PT(I+1,J,1)
                 . +LAMP*PT(I,J-1,1))
0045           15 IF (NMO.LT.2) GO TO 16
0046              DO 6 I=2,NMO
0047              PT(I,1,2)=RHS(LAMP*PT(I-1,1,1)-CLLPMP*PT(I,1,1)+MUP*PT(I+1,1,1))
0048            6 PT(I,N,2)=RHS(LAMP*PT(I-1,N,1)-CLLPMP*PT(I,N,1)+MUP*PT(I+1,N,1)
                 .  +LAM*PT(I,NMO,1))
0049           16 IF (NMO.LT.2) GO TO 17
0050              DO 7 J=2,NMO
0051              PT(1,J,2)=RHS(LAM*PT(1,J-1,1)-CLLPM*PT(1,J,1)+MU*PT(1,J+1,1)
                 .  +MUP*PT(2,J,1))
0052            7 PT(N,J,2)=RHS(LAM*PT(N,J-1,1)+LAMP*PT(NMO,J,1)-CLLPM*PT(N,J,1))
0053           17 PT(N,N,2)=RHS(LAMP*PT(NMO,N,1)-CLLPMP*PT(N,N,1)+LAM*PT(N,NMO,1))
0054              PT(1,N,2)=RHS(LAM*PT(1,NMO,1)-CLLPM*PT(1,N,1)+MUP*PT(2,N,1))
0055              PT(N,1,2)=RHS(LAMP*PT(NMO,1,1)-CLLPMP*PT(N,1,1))
0056              RETURN
                C
                C***  THE THIRD ENTRY POINT IS USED AFTER EACH TIME STEP WITH NEW
                C***  VALUES OF THE SOLUTION. AT THIS TIME ANY OTHER VARIABLES, WHICH
                C***  DEPEND ON THE P-VECTOR (EXPECTED VALUES, STANDARD DEVIATIONS,
                C***  ETC.) MAY BE COMPUTED AS NECESSARY.
                C
0057              ENTRY NEWVAL(T,PT)
0058              CALL CONTRL(IPRNT,NPRNT,621)
0059              ESYS=0.0
0060              VSYS=0.0
0061              AVQ=0.0
0062              VQ=0.0
0063              ESYSP=0.0
0064              VSYSP=0.0
```

```
0065              AVQP=0.0
0066              VQP=0.0
0067              FJ1=0.0
0068              FJ=0.0
0069              DO 8 J=1,N
0070                SUM=0.0
0071                DO 9 I=1,M
0072              9   SUM=SUM+PT(I,J,1)
0073                ESYS=ESYS+FJ*SUM
0074                VSYS=VSYS+FJ*FJ*SUM
0075                FJ=FJ+1.0
0076                IF (J.EQ.1) GO TO 8
0077                AVQ=AVQ+FJ1*SUM
0078                VQ=VQ+FJ1*FJ1*SUM
0079                FJ1=FJ1+1.0
0080              8 CONTINUE
0081                FI1=0.0
0082                FI=0.0
0083                PSUM=0.0
0084                DO 10 I=1,M
0085                  SUM=0.0
0086                  DO 11 J=1,N
0087                PSUM=PSUM+PT(M-I+1,N-J+1,1)
0088             11   SUM=SUM+PT(I,J,1)
0089                  ESYSP=ESYSP+FI*SUM
0090                  VSYSP=VSYSP+FI*FI*SUM
0091                  FI=FI+1.0
0092                  IF (I.EQ.1) GO TO 10
0093                  AVQP=AVQP+FI1*SUM
0094                  VQP=VQP+FI1*FI1*SUM
0095                  FI1=FI1+1.0
0096             10 CONTINUE
0097                VSYS=SQRT(VSYS-ESYS**2)
0098                VQ=SQRT(VQ-AVQ**2)
0099                VSYSP=SQRT(VSYSP-ESYSP**2)
0100                VQP=SQRT(VQP-AVQP**2)
0101                PSUM=100.0*(PSUM-1.0)
0102                CALL CONTRL(LINE,55,612)
0103                PRINT 3
0104             12 PRINT 4,T,ESYS,VSYS,AVQ,VQ,ESYSP,VSYSP,AVQP,VQP,PSUM
0105             21 RETURN
0106             20 CALL EXIT
                 C
                 C***  THE FOURTH ENTRY POINT IS USED TO INDICATE IF ANOTHER
                 C***  PROBLEM IS TO BE SOLVED, IN WHICH CASE EXECUTING A RETURN WILL
                 C***  CAUSE THE NEXT ENTRY TO BE MADE AT INIT; OR TO TERMINATE, IN WHICH
                 C***  CASE A STOP OR CALL EXIT COULD BE EXECUTED.
                 C
```

```
0107            ENTRY WRAPUP
0108            RETURN
0109            END
```

PREMPTIVE-RESUME MODEL WITH PRIORITY SYSTEM HAVING 2 STATES. S.OLSON APRIL 1972

LAMBDA	LAMBDA-P	MU	MU-P
0.067000	0.001670	0.333000	0.033300

T	E(SYS)	STD DEV	AV Q	STD DEV	E(SYSP)	STD DEV	AV QP	STD DEV	% ERROR
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.50	0.03089	0.17589	0.00050	0.02254	0.00083	0.02873	0.0	0.0	-0.000
1.00	0.05718	0.23385	0.00177	0.04205	0.00163	0.04034	0.0	0.0	-0.001
1.50	0.07958	0.28203	0.00356	0.06145	0.00241	0.04901	0.0	0.0	-0.003
2.00	0.09932	0.31737	0.00567	0.07822	0.00316	0.05618	0.0	0.0	-0.007
2.50	0.11674	0.34512	0.00796	0.09246	0.00387	0.06210	0.0	0.0	-0.013
3.00	0.13225	0.36777	0.01034	0.10431	0.00455	0.06733	0.0	0.0	-0.021
3.50	0.14626	0.38676	0.01274	0.11396	0.00520	0.07195	0.0	0.0	-0.031
4.00	0.15907	0.40257	0.01513	0.12165	0.00582	0.07608	0.0	0.0	-0.043
4.50	0.17087	0.41572	0.01748	0.12757	0.00641	0.07978	0.0	0.0	-0.057
5.00	0.18187	0.42674	0.01976	0.13193	0.00696	0.08314	0.0	0.0	-0.074
5.50	0.19214	0.43611	0.02197	0.13491	0.00748	0.08619	0.0	0.0	-0.092
6.00	0.20179	0.44425	0.02409	0.13669	0.00798	0.08897	0.0	0.0	-0.112
6.50	0.21089	0.45152	0.02612	0.13741	0.00845	0.09152	0.0	0.0	-0.135
7.00	0.21949	0.45822	0.02807	0.13723	0.00889	0.09386	0.0	0.0	-0.159
7.50	0.22767	0.46457	0.02993	0.13628	0.00931	0.09603	0.0	0.0	-0.185
8.00	0.23547	0.47073	0.03172	0.13468	0.00970	0.09799	0.0	0.0	-0.212
8.50	0.24298	0.47686	0.03343	0.13252	0.01006	0.09982	0.0	0.0	-0.242
9.00	0.25022	0.48306	0.03507	0.12990	0.01041	0.10151	0.0	0.0	-0.272
9.50	0.25721	0.48939	0.03665	0.12690	0.01074	0.10308	0.0	0.0	-0.304
10.00	0.26399	0.49588	0.03817	0.12358	0.01105	0.10454	0.0	0.0	-0.337
10.50	0.27057	0.50253	0.03963	0.11998	0.01134	0.10589	0.0	0.0	-0.372
11.00	0.27697	0.50933	0.04103	0.11617	0.01162	0.10715	0.0	0.0	-0.408
11.50	0.28320	0.51627	0.04237	0.11219	0.01187	0.10832	0.0	0.0	-0.444
12.00	0.28928	0.52331	0.04365	0.10808	0.01212	0.10942	0.0	0.0	-0.482
12.50	0.29522	0.53044	0.04487	0.10387	0.01235	0.11043	0.0	0.0	-0.521
13.00	0.30103	0.53762	0.04604	0.09958	0.01256	0.11139	0.0	0.0	-0.561
13.50	0.30672	0.54482	0.04716	0.09524	0.01277	0.11227	0.0	0.0	-0.601
14.00	0.31230	0.55201	0.04823	0.09087	0.01296	0.11310	0.0	0.0	-0.642
14.50	0.31778	0.55912	0.04926	0.08650	0.01314	0.11388	0.0	0.0	-0.684
15.00	0.32317	0.56614	0.05024	0.08214	0.01331	0.11461	0.0	0.0	-0.727
15.50	0.32847	0.57302	0.05118	0.07781	0.01347	0.11529	0.0	0.0	-0.771
16.00	0.33369	0.57973	0.05209	0.07352	0.01362	0.11593	0.0	0.0	-0.815
16.50	0.33883	0.58627	0.05296	0.06929	0.01377	0.11652	0.0	0.0	-0.859
17.00	0.34389	0.59266	0.05379	0.06514	0.01390	0.11708	0.0	0.0	-0.904
17.50	0.34889	0.59889	0.05459	0.06108	0.01403	0.11761	0.0	0.0	-0.950
18.00	0.35381	0.60497	0.05536	0.05714	0.01415	0.11810	0.0	0.0	-0.996
18.50	0.35868	0.61090	0.05609	0.05331	0.01426	0.11856	0.0	0.0	-1.043
19.00	0.36349	0.61669	0.05680	0.04961	0.01437	0.11899	0.0	0.0	-1.090
19.50	0.36825	0.62234	0.05748	0.04604	0.01447	0.11940	0.0	0.0	-1.137
20.00	0.37296	0.62786	0.05813	0.04260	0.01456	0.11978	0.0	0.0	-1.185

PREMPTIVE-RESUME MODEL WITH PRIORITY SYSTEM HAVING 2 STATES. S.OLSON APRIL 1972

T	P-VECTOR									
0.0	1 1.0000									
0.50	1 0.9668	2 0.0008	3 0.0299	4 0.0000	5 0.0005					
1.00	1 0.9430	2 0.0015	3 0.0536	4 0.0001	5 0.0017					
1.50	1 0.9216	2 0.0022	3 0.0725	4 0.0002	5 0.0033	6 0.0000	7 0.0001			
2.00	1 0.9037	2 0.0029	3 0.0876	4 0.0003	5 0.0052	6 0.0000	7 0.0002			
2.50	1 0.8887	2 0.0034	3 0.0998	4 0.0004	5 0.0072	6 0.0000	7 0.0004			
3.00	1 0.8759	2 0.0040	3 0.1097	4 0.0005	5 0.0091	6 0.0000	7 0.0006			
3.50	1 0.8650	2 0.0045	3 0.1176	4 0.0006	5 0.0110	6 0.0001	7 0.0008			
4.00	1 0.8557	2 0.0050	3 0.1242	4 0.0008	5 0.0128	6 0.0001	7 0.0010	8 0.0000	9 0.0001	
4.50	1 0.8477	2 0.0054	3 0.1295	4 0.0009	5 0.0145	6 0.0001	7 0.0012	8 0.0000	9 0.0001	
5.00	1 0.8407	2 0.0058	3 0.1339	4 0.0010	5 0.0161	6 0.0001	7 0.0015	8 0.0000	9 0.0001	
5.50	1 0.8347	2 0.0062	3 0.1375	4 0.0011	5 0.0175	6 0.0001	7 0.0018	8 0.0000	9 0.0001	
6.00	1 0.8294	2 0.0066	3 0.1405	4 0.0012	5 0.0188	6 0.0002	7 0.0020	8 0.0000	9 0.0002	
6.50	1 0.8247	2 0.0069	3 0.1430	4 0.0013	5 0.0200	6 0.0002	7 0.0023	8 0.0000	9 0.0002	
7.00	1 0.8206	2 0.0073	3 0.1451	4 0.0014	5 0.0211	6 0.0002	7 0.0025	8 0.0000	9 0.0002	
7.50	1 0.8169	2 0.0076	3 0.1468	4 0.0015	5 0.0221	6 0.0002	7 0.0028	8 0.0000	9 0.0003	
8.00	1 0.8136	2 0.0079	3 0.1483	4 0.0015	5 0.0230	6 0.0003	7 0.0030	8 0.0000	9 0.0003	
8.50	1 0.8106	2 0.0081	3 0.1495	4 0.0016	5 0.0238	6 0.0003	7 0.0032	8 0.0000	9 0.0004	
9.00	1 0.8079	2 0.0084	3 0.1505	4 0.0017	5 0.0245	6 0.0003	7 0.0034	8 0.0000	9 0.0004	
9.50	1 0.8055	2 0.0086	3 0.1514	4 0.0018	5 0.0252	6 0.0003	7 0.0036	8 0.0000	9 0.0004	
10.00	1 0.8033	2 0.0089	3 0.1521	4 0.0018	5 0.0258	6 0.0003	7 0.0038	8 0.0001	9 0.0005	10 0.0000
	11 0.0001									
10.50	1 0.8013	2 0.0091	3 0.1528	4 0.0019	5 0.0263	6 0.0003	7 0.0040	8 0.0001	9 0.0005	10 0.0000
	11 0.0001									
11.00	1 0.7995	2 0.0093	3 0.1533	4 0.0019	5 0.0268	6 0.0004	7 0.0041	8 0.0001	9 0.0006	10 0.0000
	11 0.0001									
11.50	1 0.7978	2 0.0095	3 0.1537	4 0.0020	5 0.0272	6 0.0004	7 0.0043	8 0.0001	9 0.0006	10 0.0000
	11 0.0001									
12.00	1 0.7962	2 0.0096	3 0.1541	4 0.0020	5 0.0276	6 0.0004	7 0.0044	8 0.0001	9 0.0006	10 0.0000
	11 0.0001									
12.50	1 0.7947	2 0.0098	3 0.1544	4 0.0021	5 0.0279	6 0.0004	7 0.0046	8 0.0001	9 0.0007	10 0.0000
	11 0.0001									
13.00	1 0.7934	2 0.0100	3 0.1547	4 0.0021	5 0.0283	6 0.0004	7 0.0047	8 0.0001	9 0.0007	10 0.0000
	11 0.0001									
13.50	1 0.7921	2 0.0101	3 0.1549	4 0.0021	5 0.0285	6 0.0004	7 0.0048	8 0.0001	9 0.0007	10 0.0000
	11 0.0001									
14.00	1 0.7909	2 0.0103	3 0.1551	4 0.0022	5 0.0288	6 0.0004	7 0.0049	8 0.0001	9 0.0008	10 0.0000
	11 0.0001									
14.50	1 0.7898	2 0.0104	3 0.1553	4 0.0022	5 0.0290	6 0.0004	7 0.0050	8 0.0001	9 0.0008	10 0.0000
	11 0.0001									
15.00	1 0.7887	2 0.0105	3 0.1554	4 0.0022	5 0.0292	6 0.0005	7 0.0051	8 0.0001	9 0.0008	10 0.0000
	11 0.0001									
15.50	1 0.7877	2 0.0106	3 0.1555	4 0.0023	5 0.0294	6 0.0005	7 0.0052	8 0.0001	9 0.0008	10 0.0000
	11 0.0001									
16.00	1 0.7868	2 0.0107	3 0.1556	4 0.0023	5 0.0296	6 0.0005	7 0.0053	8 0.0001	9 0.0009	10 0.0000
	11 0.0001									
16.50	1 0.7858	2 0.0109	3 0.1557	4 0.0023	5 0.0297	6 0.0005	7 0.0054	8 0.0001	9 0.0009	10 0.0000
	11 0.0001									
17.00	1 0.7850	2 0.0110	3 0.1557	4 0.0024	5 0.0299	6 0.0005	7 0.0054	8 0.0001	9 0.0009	10 0.0000
	11 0.0001									

PREMPTIVE-RESUME MODEL WITH PRIORITY SYSTEM HAVING 2 STATES. S.OLSON APRIL 1972

T	P-VECTOR									
17.50	1 0.7842	2 0.0110	3 0.1557	4 0.0024	5 0.0300	6 0.0005	7 0.0055	8 0.0001	9 0.0009	10 0.0000
	11 0.0001									
18.00	1 0.7834	2 0.0111	3 0.1558	4 0.0024	5 0.0301	6 0.0005	7 0.0055	8 0.0001	9 0.0010	10 0.0000
	11 0.0002									
18.50	1 0.7826	2 0.0112	3 0.1558	4 0.0024	5 0.0302	6 0.0005	7 0.0056	8 0.0001	9 0.0010	10 0.0000
	11 0.0002									
19.00	1 0.7819	2 0.0113	3 0.1558	4 0.0024	5 0.0303	6 0.0005	7 0.0056	8 0.0001	9 0.0010	10 0.0000
	11 0.0002									
19.50	1 0.7811	2 0.0114	3 0.1557	4 0.0025	5 0.0304	6 0.0005	7 0.0057	8 0.0001	9 0.0010	10 0.0000
	11 0.0002									
20.00	1 0.7805	2 0.0114	3 0.1557	4 0.0025	5 0.0304	6 0.0005	7 0.0057	8 0.0001	9 0.0010	10 0.0000
	11 0.0002									

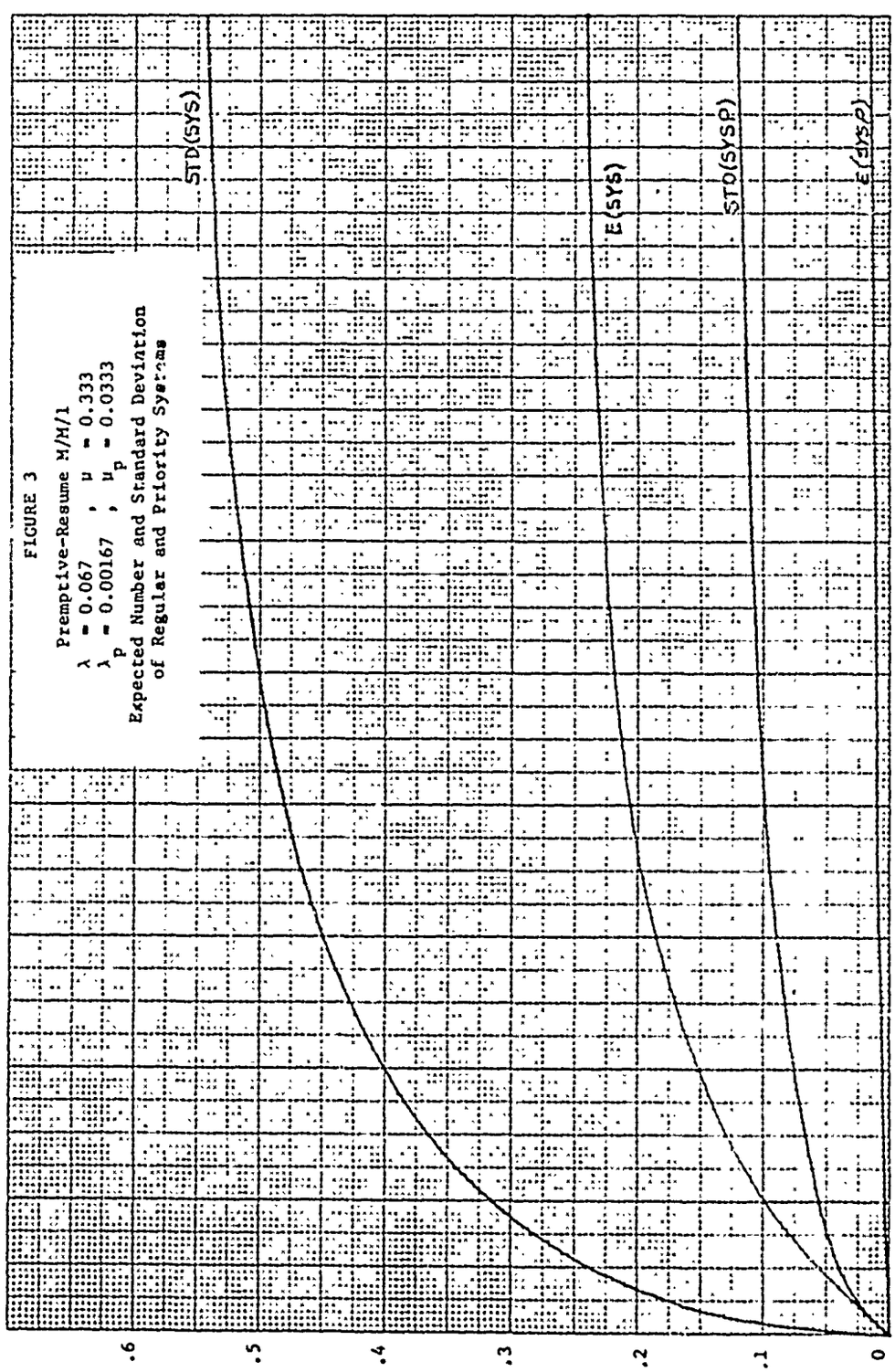


FIGURE 3
Premptive-Resume M/M/1
λ = 0.067 , μ = 0.333
λp = 0.00167 , μp = 0.0333
Expected Number and Standard Deviation
of Regular and Priority Systems
STD(SYS)
E(SYS)
STD(SYSP)
E(SYSP)
.6
.5
.4
.3
.2
.1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
t, sec

State-Dependent Queue

The model discussed here is a generalized, single-channel, state-dependent queue with s servers developed by Conway and Hillier (4). The differential equations, in terms of the independent variable $\tau = \mu t$, are

$$\dot{P}_0(\tau) = \eta_1 P_1(\tau) - \rho_0(\tau) P_0(\tau)$$

$$\dot{P}_n(\tau) = \rho_{n-1}(\tau) P_{n-1}(\tau) - [\rho_n(\tau) + \eta_n] P_n(\tau) + \eta_{n+1} P_{n+1}(\tau) ,$$

$$\text{for } 1 \leq n \leq N$$

$$\dot{P}_N(\tau) = \rho_{n-1}(\tau) P_{N-1}(\tau) - [\rho_N(\tau) + \eta_N] P_N(\tau) ,$$

where N is the maximum number of items in the system,

$$\rho_n(\tau) = \lambda_n(\tau)/\mu ,$$

$$\eta_n = \mu_n/\mu = \begin{cases} n , & n < s \\ (\frac{n}{s})^{c_s} , & n \geq s \end{cases}$$

$$\lambda_n(\tau) = \begin{cases} \lambda(\tau) , & n < s \\ \left(\frac{s}{n+1}\right)^b \lambda(\tau) , & n \geq s. \end{cases}$$

This model becomes the standard M/M/1 queue when the parameters b and c are zero and s=1. This is a useful model for working with an existing system for which empirical data are available. The time dependence of arrivals and departures can be handled as well as a wide variety of system dependence through the choice of b, c and s.

Here $\rho = \lambda/\mu$ is called the loading factor. A steady-state solution can not be obtained when $\rho > 1$. However, it is often required to study systems for which $\rho \geq 1$ during part of the time of interest. Figure 4 shows how a single channel dual server queue model behaves with $\rho = 1.5$ from $\tau = 0$ until $\tau = 8$ with the system initially empty.

Subroutine INIT for this model and sample output are included next. Note that T in the output refers to τ above.

```
FORTRAN IV G LEVEL  20                INIT              DATE = 72103          18/09/54              PAGE 0001

0001                  SUBROUTINE INIT
                      C
                      C***  STATE-DEPENDENT MODEL.
                      C***  INITIALIZATION SECTION INCLUDES COMPUTATION OF DIMENSIONLESS
                      C***  PARAMETERS WHICH ARE ONLY FUNCTIONS OF N.
                      C
                      C***  IN THIS MODEL, RHO WILL BE SET AT 1.5 FOR 0<T<8. THEN RHO WILL
                      C***  BE SET TO 0.5 TO EXPLORE HOW THE SYSTEM RECOVERS FROM OVERLOAD.
                      C***  T IS TAU IN THE EQUATIONS, THEREFORE MU IS ACCOUNTED FOR IN T.
                      C***  S=2 SERVERS WILL BE USED.
                      C
0002                  COMMON TITLE(20),TO,TMAX,DT,NPRNT,NEQ,P(21,2),W(63)
0003                  REAL PHON(21),ETAN(21),LAMBDA,MU
0004                10 READ (5,1,END=11) TITLE,TO,TMAX,DT,NPRNT,NEQ,RHO1,RHO2,B,C,S
0005                 1 FORMAT (20A4/3F10.0,5X2I5/5F10.0)
0006                   N=NEQ-1
0007                   P(1,1)=1.0
0008                   DO 2 I=2,NEQ
0009                 2 P(I,1)=0.0
                      C
                      C***  COMPUTE FUNCTIONS ONLY OF N.
                      C
0010                   ETAN(1)=1.0
0011                   RHON(1)=1.0
0012                   FS=S
0013                   FN=1.0
0014                   DO 3 I=2,NEQ
0015                   F2=FN+1.0
0016                   IF (FN.GE.FS) GO TO 4
0017                   RHON(I)=1.0
0018                   ETAN(I)=FN
0019                   GO TO 3
0020                 4 ETAN(I)=(FN/FS)**C*S
0021                   RHON(I)=(FS/F2)**B
0022                 3 FN=F2
0023                   WRITE (6,5) TITLE,TO,TMAX,DT,NPRNT,NEQ,RHO1,RHO2,B,C,S
0024                 5 FORMAT (1H1/20A4//1H0,13X,2HTO,11X,4HTMAX,13X,2HDT,10X,5HNPRNT,12X,3HNEQ/
                     . 1H ,3F15.4,2I15//1H0,7X,8HRHO(T<8),7X,8HRHO(T>8),14X,1HB,14X,1HC,14X,1HS
                     . /1H ,5F15.4)
0025                   LINE=55
0026                   IPRNT=0
0027                   RETURN
                      C
0028                   ENTRY PRIME(T,PV,K)
0029                   DIMENSION PV(21,2)
0030                   IF (T.LE.8.0) RHO=RHO1
0031                   IF (T.GT.8.0) RHO=RHO2
0032                   PV(1,2)=RHS(ETAN(2)*PV(2,1)-RHON(1)*RHO*PV(1,1))
```

```
0033              DO 7 I=2,N
0034            7 PV(I,2)=RHS(ETAN(I+1)*PV(I+1,1)-(ETAN(I)+RHON(I)*RHO)*PV(I,1)
                 . +RHON(I-1)*RHO*PV(I-1,1))
0035              PV(NEQ,2)=RHS(RHON(NEQ)*RHO*P(NEQ,1)-(ETAN(N)+RHON(N)*RHO)*
                 . PV(N,1))
0036              RETURN
                C
0037              ENTRY NEWVAL(T,PV)
0038            8 CALL CONTRL(IPRNT,NPRNT,69)
0039              ERROR=PV(1,1)
0040              ESYS=0.0
0041              STDSYS=0.0
0042              FJ=1.0
0043              DO 12 J=2,NEQ
0044              ERROR=ERROR+PV(J,1)
0045              ESYS=ESYS+FJ*PV(J,1)
0046              STDSYS=STDSYS+FJ*FJ*PV(J,1)
0047           12 FJ=FJ+1.0
0048              ERROR=100.0*(ERROR-1.0)
0049              STDSYS=SQRT(ABS(STDSYS-ESYS**2))
0050              CALL CONTRL(LINE,55,13)
0051              WRITE (6,14) TITLE
0052           14 FORMAT (1H1/20A4//1HO 14X1HT 11X4HESYS 9X6HSTDSYS 3X12HPERCENT ERR.)
0053           13 WRITE (6,15) T,ESYS,STDSYS,ERROR
0054           15 FORMAT (1H F15.2,3F15.6)
0055            9 RETURN
                C
0056              ENTRY WRAPUP
0057              RETURN
0058           11 CALL EXIT
0059              END
```

STATE-DEPENDENT QUEUE S.OLSON 56-299 A.RIL,1972

TO	TMAX	DT	NPRNT	NEQ
0.0	20.0000	0.1000	5	21

RHO1(Y<B)	RHO(T>B)	B	C	S
1.5000	0.5000	0.0	0.0	2.0000

STATE-DEPENDENT QUEUE S.OLSON 56-299 APRIL,1972

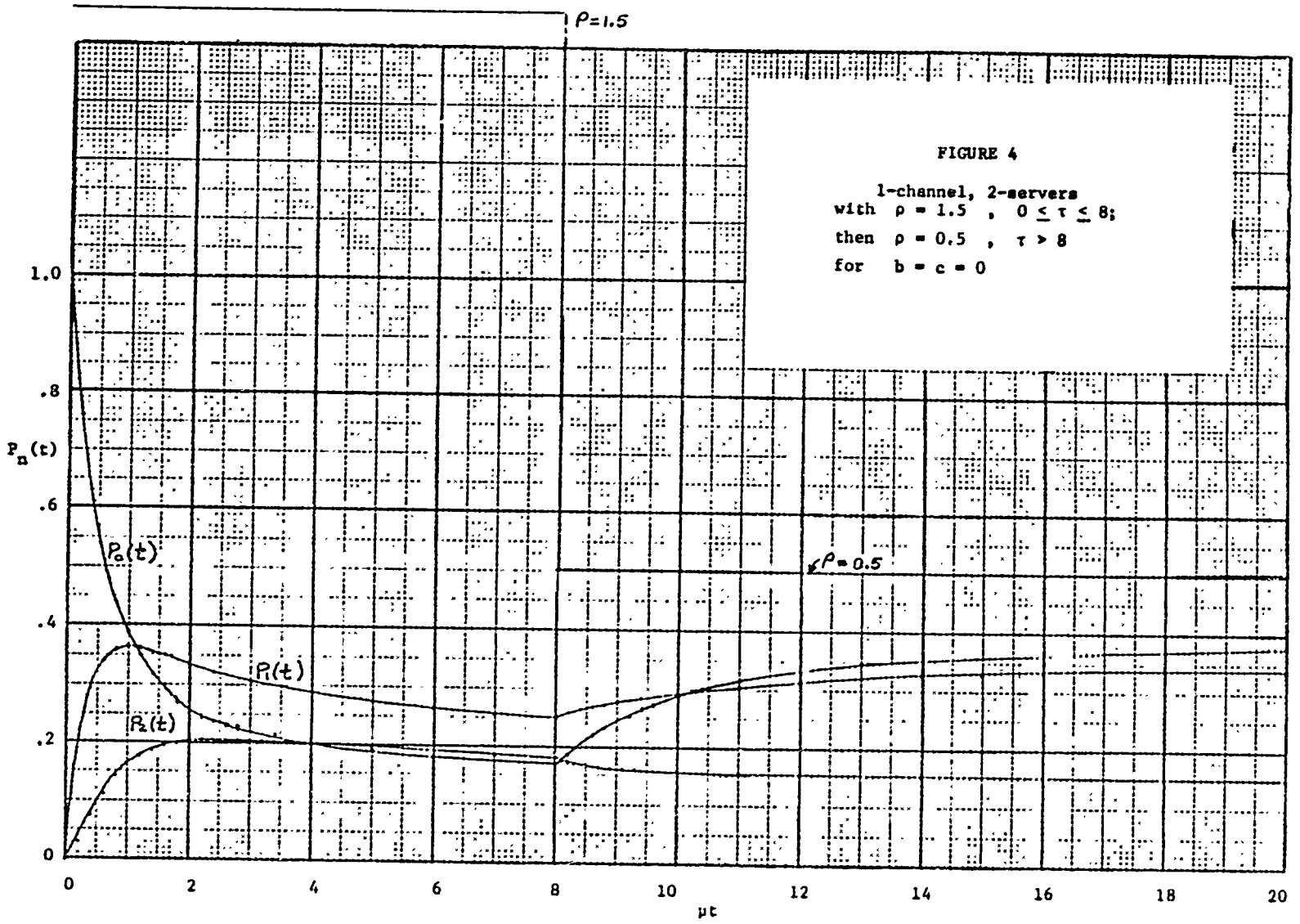
T	FSYS	STDSYS	PERCENT ERR.
0.50	0.594084	0.778937	-0.000024
1.00	0.979432	1.033791	-0.000072
1.50	1.252798	1.217929	-0.000125
2.00	1.461122	1.368942	-0.000185
2.50	1.628190	1.498303	-0.000250
3.00	1.767680	1.611950	-0.000322
3.50	1.885562	1.713208	-0.000393
4.00	1.988569	1.804424	-0.000501
4.50	2.075423	1.887289	-0.000602
5.00	2.166457	1.963672	-0.000799
5.50	2.233365	2.032743	-0.001121
6.00	2.299421	2.097050	-0.001657
6.50	2.359611	2.156566	-0.002581
7.00	2.414698	2.211755	-0.003999
7.50	2.465294	2.263001	-0.006092
8.00	2.511992	2.310629	-0.008976
8.50	2.117857	2.226742	-0.010633
9.00	1.790101	2.103646	-0.010502
9.50	1.528924	1.965032	-0.009739
10.00	1.321059	1.822382	-0.009125
10.50	1.155417	1.684881	-0.008947
11.00	1.024432	1.555268	-0.009120
11.50	0.920645	1.436865	-0.009412
12.00	0.838336	1.330827	-0.009650
12.50	0.773279	1.237668	-0.009787
13.00	0.721919	1.156974	-0.009817
13.50	0.681413	1.087961	-0.009817
14.00	0.649496	1.029609	-0.009823
14.50	0.624371	0.980802	-0.009882
15.00	0.604612	0.940402	-0.009948
15.50	0.589090	0.907294	-0.010078
16.00	0.576909	0.880407	-0.010031
16.50	0.567359	0.858749	-0.010097
17.00	0.559879	0.841420	-0.010145
17.50	0.554025	0.827634	-0.010175
18.00	0.549448	0.816714	-0.010210
18.50	0.545872	0.808101	-0.010258
19.00	0.543079	0.801331	-0.010300
19.50	0.540901	0.796025	-0.010353
20.00	0.539203	0.791878	-0.010377

STATE-DEPENDENT QUEUE S.OLSON 56-299 APRIL,1972

T	P-VECTOR																			
0.0	1 1.0000																			
0.50	1 0.5542	2 0.3267	3 0.0947	4 0.0204	5 0.0035	6 0.0005	7 0.0001													
1.00	1 0.3068	2 0.3637	3 0.1649	4 0.0603	5 0.0183	6 0.0047	7 0.0011	8 0.0002												
1.50	1 0.3030	2 0.3531	3 0.1935	4 0.0902	5 0.0362	6 0.0127	7 0.0039	8 0.0011	9 0.0003	10 0.0001										
2.00	1 0.2668	2 0.3361	3 0.2033	4 0.1085	5 0.0513	6 0.0217	7 0.0082	8 0.0028	9 0.0009	10 0.0003										
	11 0.0001																			
2.50	1 0.2411	2 0.3285	3 0.2054	4 0.1193	5 0.0627	6 0.0299	7 0.0130	8 0.0052	9 0.0019	10 0.0006										
	11 0.0002	12 0.0001																		
3.00	1 0.2240	2 0.3074	3 0.2045	4 0.1255	5 0.0709	6 0.0369	7 0.0177	8 0.0079	9 0.0033	10 0.0012										
	11 0.0004	12 0.0002																		
3.50	1 0.2117	2 0.2967	3 0.2023	4 0.1291	5 0.0768	6 0.0426	7 0.0220	8 0.0106	9 0.0048	10 0.0020										
	11 0.0008	12 0.0003	13 0.0001																	
4.00	1 0.2026	2 0.2878	3 0.1997	4 0.1311	5 0.0811	6 0.0472	7 0.0258	8 0.0133	9 0.0064	10 0.0029										
	11 0.0013	12 0.0005	13 0.0002	14 0.0001																
4.50	1 0.1954	2 0.2804	3 0.1972	4 0.1322	5 0.0842	6 0.0508	7 0.0290	8 0.0157	9 0.0081	10 0.0039										
	11 0.0018	12 0.0007	13 0.0003	14 0.0001																
5.00	1 0.1897	2 0.2742	3 0.1947	4 0.1327	5 0.0864	6 0.0537	7 0.0318	8 0.0179	9 0.0096	10 0.0049										
	11 0.0024	12 0.0011	13 0.0005	14 0.0002	15 0.0001															
5.50	1 0.1850	2 0.2688	3 0.1924	4 0.1328	5 0.0881	6 0.0561	7 0.0342	8 0.0199	9 0.0111	10 0.0059										
	11 0.0030	12 0.0015	13 0.0007	14 0.0003	15 0.0001	16 0.0001														
6.00	1 0.1810	2 0.2642	3 0.1902	4 0.1327	5 0.0894	6 0.0580	7 0.0362	8 0.0217	9 0.0125	10 0.0069										
	11 0.0037	12 0.0019	13 0.0009	14 0.0004	15 0.0002	16 0.0001														
6.50	1 0.1776	2 0.2602	3 0.1883	4 0.1325	5 0.0904	6 0.0596	7 0.0379	8 0.0233	9 0.0137	10 0.0078										
	11 0.0043	12 0.0023	13 0.0011	14 0.0005	15 0.0003	16 0.0001	17 0.0001													
7.00	1 0.1747	2 0.2567	3 0.1865	4 0.1322	5 0.0911	6 0.0609	7 0.0394	8 0.0247	9 0.0149	10 0.0087										
	11 0.0049	12 0.0027	13 0.0014	14 0.0006	15 0.0003	16 0.0002	17 0.0001													
7.50	1 0.1722	2 0.2536	3 0.1849	4 0.1318	5 0.0916	6 0.0619	7 0.0407	8 0.0259	9 0.0160	10 0.0095										
	11 0.0055	12 0.0031	13 0.0017	14 0.0008	15 0.0004	16 0.0002	17 0.0001													
8.00	1 0.1700	2 0.2508	3 0.1834	4 0.1314	5 0.0920	6 0.0628	7 0.0418	8 0.0270	9 0.0169	10 0.0103										
	11 0.0061	12 0.0035	13 0.0019	14 0.0010	15 0.0005	16 0.0003	17 0.0001	18 0.0001												
8.50	1 0.2479	2 0.2737	3 0.1579	4 0.1034	5 0.0750	6 0.0508	7 0.0335	8 0.0215	9 0.0134	10 0.0081										
	11 0.0048	12 0.0027	13 0.0015	14 0.0008	15 0.0004	16 0.0002	17 0.0001													
9.00	1 0.3165	2 0.2836	3 0.1405	4 0.0898	5 0.0608	6 0.0407	7 0.0266	8 0.0170	9 0.0105	10 0.0063										
	11 0.0037	12 0.0021	13 0.0012	14 0.0006	15 0.0003	16 0.0002	17 0.0001													
9.50	1 0.3728	2 0.2988	3 0.1273	4 0.0752	5 0.0493	6 0.0326	7 0.0211	8 0.0134	9 0.0082	10 0.0049										
	11 0.0029	12 0.0016	13 0.0009	14 0.0005	15 0.0002	16 0.0001														
10.00	1 0.4189	2 0.3126	3 0.1168	4 0.0637	5 0.0401	6 0.0261	7 0.0167	8 0.0105	9 0.0064	10 0.0038										
	11 0.0022	12 0.0013	13 0.0007	14 0.0004	15 0.0002	16 0.0001														
10.50	1 0.4553	2 0.3241	3 0.1084	4 0.0545	5 0.0328	6 0.0209	7 0.0133	8 0.0083	9 0.0050	10 0.0030										
	11 0.0017	12 0.0010	13 0.0005	14 0.0003	15 0.0001	16 0.0001														
11.00	1 0.4856	2 0.3357	3 0.1017	4 0.0471	5 0.0270	6 0.0167	7 0.0105	8 0.0065	9 0.0039	10 0.0023										
	11 0.0013	12 0.0007	13 0.0004	14 0.0002	15 0.0001															
11.50	1 0.5092	2 0.3468	3 0.0963	4 0.0413	5 0.0223	6 0.0135	7 0.0083	8 0.0051	9 0.0031	10 0.0018										
	11 0.0010	12 0.0006	13 0.0003	14 0.0002	15 0.0001															
12.00	1 0.5281	2 0.3575	3 0.0919	4 0.0366	5 0.0186	6 0.0109	7 0.0066	8 0.0040	9 0.0024	10 0.0014										
	11 0.0008	12 0.0004	13 0.0002	14 0.0001	15 0.0001															
12.50	1 0.5431	2 0.3683	3 0.0885	4 0.0329	5 0.0157	6 0.0088	7 0.0052	8 0.0032	9 0.0019	10 0.0011										
	11 0.0006	12 0.0003	13 0.0002	14 0.0001																
13.00	1 0.5556	2 0.3787	3 0.0857	4 0.0299	5 0.0134	6 0.0072	7 0.0042	8 0.0025	9 0.0015	10 0.0008										
	11 0.0005	12 0.0002	13 0.0001	14 0.0001																

STATE-DEPENDENT QUEUE S.OLSON 56-299 APRIL,1972

T	P-VECTOR									
13.50	1 0.5645	2 0.2991	3 0.0835	4 0.0276	5 0.0115	6 0.0059	7 0.0033	8 0.0020	9 0.0011	10 0.0007
	11 0.0004	12 0.0002	13 0.0001	14 0.0001						
14.00	1 0.5720	2 0.2993	3 0.0817	4 0.0257	5 0.0101	6 0.0049	7 0.0027	8 0.0015	9 0.0009	10 0.0005
	11 0.0003	12 0.0002	13 0.0001							
14.50	1 0.5779	2 0.2995	3 0.0803	4 0.0243	5 0.0089	6 0.0041	7 0.0022	8 0.0012	9 0.0007	10 0.0004
	11 0.0002	12 0.0001	13 0.0001							
15.00	1 0.5826	2 0.2996	3 0.0792	4 0.0231	5 0.0080	6 0.0034	7 0.0017	8 0.0010	9 0.0005	10 0.0003
	11 0.0002	12 0.0001								
15.50	1 0.5863	2 0.2997	3 0.0783	4 0.0222	5 0.0073	6 0.0030	7 0.0014	8 0.0008	9 0.0004	10 0.0002
	11 0.0001	12 0.0001								
16.00	1 0.5893	2 0.2998	3 0.0776	4 0.0214	5 0.0067	6 0.0026	7 0.0012	8 0.0006	9 0.0003	10 0.0002
	11 0.0001	12 0.0001								
16.50	1 0.5916	2 0.2999	3 0.0770	4 0.0208	5 0.0063	6 0.0023	7 0.0010	8 0.0005	9 0.0003	10 0.0001
	11 0.0001									
17.00	1 0.5934	2 0.2999	3 0.0766	4 0.0204	5 0.0059	6 0.0020	7 0.0008	8 0.0004	9 0.0002	10 0.0001
	11 0.0001									
17.50	1 0.5948	2 0.2999	3 0.0763	4 0.0200	5 0.0057	6 0.0018	7 0.0007	8 0.0003	9 0.0002	10 0.0001
18.00	1 0.5959	2 0.2999	3 0.0760	4 0.0198	5 0.0055	6 0.0017	7 0.0006	8 0.0003	9 0.0001	10 0.0001
18.50	1 0.5968	2 0.2999	3 0.0758	4 0.0195	5 0.0053	6 0.0016	7 0.0005	8 0.0002	9 0.0001	10 0.0001
19.00	1 0.5975	2 0.3000	3 0.0756	4 0.0194	5 0.0052	6 0.0015	7 0.0005	8 0.0002	9 0.0001	
19.50	1 0.5980	2 0.3000	3 0.0755	4 0.0192	5 0.0051	6 0.0014	7 0.0004	8 0.0002	9 0.0001	
20.00	1 0.5984	2 0.3000	3 0.0754	4 0.0191	5 0.0050	6 0.0014	7 0.0004	8 0.0001	9 0.0001	


ρ=1.5
FIGURE 4
1-channel, 2-servers
with ρ = 1.5 , 0 ≤ τ ≤ 8;
then ρ = 0.5 , τ > 8
for b = c = 0
1.0
.8
.6
.4
.2
0
$P_n(t)$
$P_0(t)$
$P_1(t)$
$P_2(t)$
ρ=0.5
0
2
4
6
8
10
12
14
16
18
20
μt

CHAPTER 5

SUMMARY

Methods for transient solutions of queueing problems have been explored and evaluated for their usefulness in numerical work. The so-called closed-form analytic solutions are the least-preferred forms for numerical work based on difficulty of analysis, computer programming and cost. Monte-Carlo simulation is easier and cheaper to apply than these analytic solutions but should not be used when the queueing model can be expressed in the form of differential-difference equations. In this case, the use of Runge-Kutta integration is preferred.

The main control program in Appendix C has been shown to be easy to use, capable of high precision, low in cost and useful in a wide range of queueing models. It is offered as a much-needed tool to researchers and practitioners in queueing theory with the hope that the use of transient analyses in queueing theory will be more easily attainable. To this extent, existing results will be made more meaningful (2).

Two areas of further research have become apparent during the preparation of this thesis. One concerns the incorporation of the Runge-Kutta queueing problem solver

described in Appendix C in continuous optimization applications. The other concerns numerical analysis on the Runge-Kutta algorithm as employed here.

The ability to obtain fast and inexpensive transient solutions to queueing problems suggests the application of variational methods to compute optimal functional solutions rather than simple scalar solutions. An example might be a problem of finding the optimal service rate of a system subject to a given fluctuating rate of arrivals and to constraints on the maximum number in the system. The optimality conditions might include a cost function relating how cost varies as a function of service rate or capacity, number in the system and the loss of arrivals when the system is full. The optimizing equation, representing the integral of the variational of the optimality conditions would then have the solution of the queueing model imbedded within its solution. A new main program to solve the queueing problem as a part of the optimization solution would be required in addition to what is presented in this thesis. Numerous optimization algorithms are available for ready incorporation in such a program.

Additional work in the numerical analysis of the Runge-Kutta algorithm was suggested in Chapter two. The Runge-Kutta algorithm can be derived in many forms depending upon the desired error in a single step and assumptions regarding the behavior of the equations being integrated. The form

employed in this thesis is one commonly-used on a great variety of problems. It can be made to work in most difficult cases by decreasing the step size until stability is eventually achieved. Generally, when this occurs, the analyst realizes the algorithm may not be appropriate and seeks another more suited to the problem.

On the other hand, it has been discovered that the Runge-Kutta equations will solve the queueing models in this thesis accurately with very large step sizes. This would seem to indicate that the method is more accurate than it need be. The differential equations in queueing are a specialized type of problem. They are usually first-order, highly-coupled and yet stable under drastically-changing conditions of the rate parameters. Future work should attempt to rederive the Runge-Kutta formula for the queueing equations. It may be possible to obtain a more economical form utilizing the characteristics of queueing models and with fewer evaluations of the differential equations at each step.

LIST OF REFERENCES

1. Beckett, R. and Hurt, J., Numerical Calculations and Algorithms, McGraw-Hill Book Company, Inc., New York, 1967.

2. Bhat, U.N., "Sixty years of queueing theory", Management Science, Vol. 15, No. 6, p.B280-94, 1969.

3. Clark, A.B., "The time-dependent waiting line problem", University of Michigan Engineering Research Institute Report No. M720-1 R 39, 1953.

4. Conway, R.W. and Hillier, F.S., "A multiple server queueing model with state dependent service rate", Journal of Industrial Engineering, Vol. XV, No. 3, 1964.

5. Gue, R.L. and Thomas, M.E., Mathematical Methods in Operations Research, The Macmillan Company, New York, 1968.

6. "IBM Operating System 360: FORTRAN IV Language", IBM Systems Reference Library, C28-6515, 1968.

7. "IBM Operating System 360: Scientific Subroutine Package", IBM Systems Reference Library, H20-0205, 1966.

8. Kaspar, H., "Methods for a transient solution to a queueing problem", TRW Systems Group, Redondo Beach, Cal., 1971.

9. Kiviat, P.J., Villanueva, R. and Markowitz, H., The SIMSCRIPT-II Programming Language, Prentice-Hall, Englewood Cliffs, N.J., 1968.

10. Leese, E.L. and Boyd, D.W., "Numerical methods of determining the transient behavior of queues with variable arrival times", Canadian Operational Research Journal, Vol. 4, No. 1, 1966.

11.Neuts, M.F., "The single server queue in discrete time - numerical analysis I", Mimeograph Series No. 270, Dept. of Statistics, Purdue University, Nov., 1971.

12.Newell, G.F., "Queues with time-dependent arrival rates: I-The transition thru saturation", Journal of Applied Probability, Vol. 5, p.436, 1968.

13._____, "Queues with time-dependent arrival rates: II-The maximum queue and the return to equilibrium", Journal of Applied Probability, Vol. 5, p.579, 1968.

14.Nielsen, K.L., College Outline Series-Differential Equations, Barnes and Noble, Inc., New York, 1962.

15.Saaty, T.L., "Resume of useful formulas in queueing theory", Operations Research, Vol. 5, p.161, 1957.

16.Schlenker, G.J., "Numerical methods in renewal theory", Master's Thesis, Dept. of Industrial and Management Engineering, University of Iowa, Feb , 1968.

GENERAL REFERENCES

1. Ackoff, R.L., ed., Progress in Operations Research, Volume I, John Wiley and Sons, Inc., New York, 1961.

2. Arora, K.L., "Time-dependent solution of the two-server queue fed by general normal and exponential service time distribution", Operations Research, Vol. 10, No. 3, 1962.

3. Bagchi, T.P. and Subramanian, V., "Bayesian approach to the design of [illegible]", Canadian Journal of Operational Research and Information Processing, Vol. 10, No. 1, 1972.

4. Bailey, N.TJ., The Elements of Stochastic Processes with Applications to the Natural Sciences, John Wiley and Sons, Inc., New York, 1964.

5. Finch, P.D., "On the transient behavior of a simple queue", Journal of the Royal Statistical Society, B22, p.277, 1960.

6. Galliher, H.P. and Wheeler, R.C., "Non-stationary queueing probabilities for landing congestion of aircraft", Operations Research, Vol. 6, p.264, 1958.

7. Gnedenko, B.V. and Kovalenko, I.N., Introduction to Queueing Theory, Weizmann Science Press Limited, Jerusalem, Israel, 1968.

8. Greenberg, I., "Some [illegible] in the theory of queues", Journal of Applied Probability, Vol. 6, p.99, 1969.

9. Heyne, J.B. and Brotman, B., "On pulse-type arrivals to a queue", Operations Research, Vol. 8, No. 3, 1960.

10. Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, Holden-Day, Inc., San Francisco, Cal., 1968.

11. Jaiswal, N.K., Priority Queues, Academic Press, New York, 1968.

12. Luchak, G., "The solution of the single-channel queueing equations characterized by a time-dependent arrival rate and a general class of holding times", Operations Research, Vol. 4, p.711, 1956.

13. _____, "The distribution of the time required to reduce to some preassigned level a single-channel queue characterized by a time-dependent Poisson-distributed arrival rate and a general class of holding times", Operations Research, Vol. 5, p.205, 1957.

14. Morse, P.M., Queues, Inventories and Maintenance, John Wiley and Sons, Inc., New York, 1958.

15. Saaty, T.L., "Time-dependent solution of the many-server Poisson queue", Operations Research, Vol. 8, No. 6, 1960.

16. _____, Elements of Queueing Theory, McGraw-Hill Book Company, Inc., New York, 1961.

17. Takacs, L., "Transient behavior of a single-channel, single-server queueing process with recurrent input and exponentially distributed service times", Operations Research, Vol. 8, p.321, 1960.

18. _____, Introduction to the Theory of Queues, Oxford University Press, New York, 1962.

APPENDIX A

LISTING OF MONTE-CARLO SIMULATION PROGRAM

The computer program listed here is a simulation of a M/M/1 queue with the same rate parameters and maximum number of items in the system N as used in the comparison in Chapter three. The operation of the simulation is straightforward and largely self-explanatory due to the uniqueness of the SIMSCRIPT-II language. The elements of the simulation include a QUEUE which is FIFO by default, ARRIVAL events which cause ITEMS to be created and SERVICE events. When an ARRIVAL event occurs, an ITEM is created, its time of arrival is saved and it is filed in the QUEUE. The time between arrivals is an exponentially-distributed random variable computed by the EXPONENTIAL.F function. ARRIVALS reschedule themselves. A SERVICE event is triggered whenever an ITEM joins the empty QUEUE and/or when an ITEM completes SERVICE and the QUEUE is not empty. The SERVICE completion event time is also scheduled using exponentially-distributed random variables.

At pre-specified times during the time the simulation is allowed to run, events named LOOK occur. This event records the instantaneous number in the system in a histogram named P. The simulation also computes average

waiting time in the system and percent utilization of the server.

The simulation is replicated 1000 times while the histogram is accumulated. The histogram cells are then divided by 1000 to give estimates of the probability of the number in the system at a particular time and printed. The sample output shows this for times of 5, 10, 15 and 20. These times correspond to times printed in the other two methods discussed in Chapter three. The results are in fair agreement with the Runge-Kutta solution in Chapter three.

```
LINE  CACI SIMSCRIPT II.5  VERSION 5                    04/12/72  PAGE   1

   1  PREAMBLE.....
   2
   3        NORMALLY, MODE IS REAL AND DIMENSION IS 0
   4
   5        PERMANENT ENTITIES...
   6            THE SYSTEM OWNS A QUEUE
   7            EVERY SNAP HAS A T.SNAP, AN AVSYS AND A STOSYS.
   8
   9        TEMPORARY ENTITIES...
  10            EVERY ITEM HAS A T.ARRVL AND MAY BELONG TO SOME QUEUE.
  11
  12        EVENT NOTICES INCLUDE ARRIVAL, SERVICE AND LOOK.
  13        PRIORITY ORDER IS ARRIVAL, SERVICE AND LOOK
  14
  15        DEFINE BUSY.SW,NMAX,NREPS,REP,N,NITEM,SITEM,DEBUG,TOTAL
  16                AS INTEGER VARIABLES
  17        DEFINE P AS A 2-DIMENSIONAL ARRAY.
  18        DEFINE ARR.RATE,SERV.RATE,MTBA,MTBS,TEND,T.BUSY,T.WAIT,T.START,
  19            TSAVE,TCHANGE AS VARIABLES
  20        DEFINE LAZY TO MEAN DEFINE X AS AN INTEGER VARIABLE.
  21
  22  END...
```

```
    1  MAIN.....
    2        LAZY
    3
    4  'NEXTCASE'
    5        IF DATA IS ENDED STOP
    6        OTHERWISE
    7
    8        READ ARR.RATE,SERV.RATE,NMAX,N.SNAP,NREPS AND DEBUG
    9        LET N=NMAX+1
   10        CREATE EVERY SNAP
   11        RESERVE P AS N BY N.SNAP
   12        FOR EACH SNAP, READ T.SNAP(SNAP)
   13        LET TEND=T.SNAP(N.SNAP)
   14        FOR EACH SNAP, DO
   15            LET AVSYS(SNAP)=0.0
   16            LET STDSYS(SNAP)=0.0
   17            FOR I=1 TO N, LET P(I,SNAP)=0.0
   18        LOOP
   19        LET MIDA=1.0/ARR.RATE
   20        LET MIBS=1.0/SERV.RATE
   21        LET T.BUSY=0.0
   22        LET T.WAIT=0.0
   23        LET TOTAL=0
   24
   25        FOR REP=1 TO NREPS, DO
   26            LET TIME.V=0.0
   27            LET SNAP=1
   28            LET SITEM=0
   29            LET TSAVE=0.0
   30            LET TCHANGE=0.0
   31            LET NITEM=0
   32            SCHEDULE AN ARRIVAL NEXT
   33            SCHEDULE A LOOK AT T.SNAP(SNAP)
   34            LET BUSY.SW=0
   35
   36            START SIMULATION
   37
   38  'AGAIN'   IF QUEUE IS NOT EMPTY,
   39                 REMOVE THE FIRST ITEM FROM QUEUE
   40                 SUBTRACT 1 FROM NITEM
   41                 DESTROY THIS ITEM
   42                 GO AGAIN
   43            OTHERWISE
   44            SUBTRACT BUSY.SW FROM NITEM
   45        ADD NITEM TO TOTAL
   46        LOOP
   47        START NEW PAGE
   48        PRINT 4 LINES WITH ARR.RATE,SERV.RATE,NMAX,NREPS,
   49             T.BUSY/TIME.V/NREPS*100.0 AND T.WAIT/TOTAL AS FOLLOWS
SIMULATION OF SINGLE-CHANNEL, SINGLE-SERVER QUEUE  S.OLSON,56-299,APRIL,1972

ARR. RATE SER. RATE QUE. LIMIT  REPLICATIONS  % SERV. UTILIZATION  AVG. WAIT
  **.***    **.***      ***         *****          ***.****         ****.***
```

```
 LINE  CACI SIMSCRIPT II.5  VERSION 5                    04/12/72  PAGE   3

   50
   51         FOR EACH SNAP, DO
   52             LET AVG=AVSYS(SNAP)/NREPS
   53             LET STD=SQRT.F(ABS.F(STDSYS(SNAP)-NREPS*AVG*AVG)/(NREPS-1))
   54             START NEW PAGE
   55             PRINT 2 LINES WITH T.SNAP(SNAP), AVG AND STD AS FOLLOWS
TIME-----.****  AV.NO. IN SYS.=****.***   STD.DEV.=****.***

   56             FOR I=1 TO N, PRINT 1 LINE WITH I-1,P(I,SNAP)/NREPS AS FOLLOWS
P(**)=*.******
   57
   58         LOOP
   59         RELEASE P,T.SNAP,AVSYS AND STDSYS
   60         GO TO NEXTCASE
   61  END.....
```

```
LINE  CACI SIMSCRIPT II.5  VERSION 5                    04/12/72  PAGE   4

   1  UPON ARRIVAL SAVING THE EVENT NOTICE...
   2
   3        LET TNEXT=TIME.V+EXPONENTIAL.F(MTBA,1)
   4        IF TNEXT IS GREATER THAN TEND
   5            DESTROY THIS ARRIVAL
   6            GO JOIN
   7        OTHERWISE
   8        RESCHEDULE THIS ARRIVAL AT TNEXT
   9  'JOIN'
  10        IF N.QUEUE EQUALS NMAX RETURN
  11        OTHERWISE
  12        CREATE AN ITEM
  13        LET T.ARRVL(ITEM)=TIME.V
  14        IF BUSY.SW IS ZERO
  15            SCHEDULE A SERVICE NEXT
  16        REGARDLESS
  17        FILE THE ITEM IN QUEUE
  18        ADD 1 TO NITEM
  19        LET NSYS=N.QUEUE+BUSY.SW
  20        ADD NSYS*(TIME.V-TCHANGE) TO TSAVE
  21        LET TCHANGE=TIME.V
  22
  23        IF DEBUG IS ZERO RETURN
  24        OTHERWISE
  25        PRINT 1 LINE WITH TIME.V, NITEM,N.QUEUE THUS
AT TIME ***.***, ITEM *** JOINS QUEUE; LENGTH=*** ----------------------ARRIVAL
  26        RETURN
  27    END.....
```

```
 LINE  CACI SIMSCRIPT II.5  VERSION 5                          04/12/72  PAGE   5

    1  UPON SERVICE SAVING THE EVENT NOTICE...
    2
    3        LET NSYS=N.QUEUE+BUSY.SW
    4        ADD NSYS*(TIME.V-TCHANGE) TO TSAVE
    5        LET TCHANGE=TIME.V
    6            LET ELAPSED=TIME.V-T.START
    7            ADD ELAPSED TO T.BUSY
    8            ADD ELAPSED TO T.WAIT
    9
   10        IF QUEUE IS EMPTY,
   11            DESTROY THIS SERVICE
   12            LET BUSY.SW=0
   13            RETURN
   14      . OTHERWISE
   15
   16        IF DEBUG IS ZERO GO NORMALLY
   17        OTHERWISE
   18        ADD 1 TO SITEM
   19        PRINT 1 LINE WITH TIME.V,SITEM,N.QUEUE AS FOLLOWS
AT TIME ***.***, ITEM *** BEGINS SERVICE, LENGTH=*** ----------------------SERVICE
   20  'NORMALLY'
   21
   22        REMOVE THE FIRST ITEM FROM QUEUE
   23        ACC TIME.V-T.ARRVL(ITEM) TO T.WAIT
   24        DESTROY THIS ITEM
   25        LET TNEXT=TIME.V+EXPONENTIAL.F(MTBS,2)
   26        LET BUSY.SW=1
   27        LET T.START=TIME.V
   28        IF TNEXT IS GREATER THAN TEND
   29            DESTROY THIS SERVICE
   30            RETURN
   31        OTHERWISE
   32        RESCHEDULE THIS SERVICE AT TNEXT
   33        RETURN
   34  END.....
```

```
 LINE  CACI SIMSCRIPT II.5  VERSION 5                          04/12/72  PAGE   6

    1  UPON LOOK SAVING THE EVENT NOTICE...
    2
    3        LAZY
    4        IF SNAP EQUALS N.SNAP
    5            DESTROY THIS LOOK
    6            GO RECORD
    7        OTHERWISE
    8        RESCHEDULE THIS LOOK AT T.SNAP(SNAP+1)
    9  *RECORD*
   10        LET I=N.QUEUE(1,BUSY,SW
   11        ADD I TO AVSYS(SNAP)
   12        ADD I*I TO SUMSYS(SNAP)
   13        ADD 1.0 TO P(I+1,SNAP)
   14        ADD 1 TO SNAP
   15
   16        IF DEBUG IS ZERO RETURN
   17        OTHERWISE
   18        PRINT 1 LINE WITH TIME.V,SNAP-1,I AS FOLLOWS
AT TIME ***.***, SNAPSHOT *** WITH *** IN SYSTEM ----------------------------LOOK
   19
   20        RETURN
   21  END.....
```

SIMULATION OF SINGLE-CHANNEL, SINGLE-SERVER QUEUE S.OLSON,56-299,APRIL,1972

ARR. RATE	SER. RATE	QUE. LIMIT	REPLICATIONS	% SERV. UTILIZATION	AVG. WAIT
.450	.500	20	1000	13.1473	2.501

```
TIME=  5.0000  AV.NO. IN SYS.=  1.565   STD.DEV.=  1.541

P( 0)= .301000
P( 1)= .264000
P( 2)= .196000
P( 3)= .127000
P( 4)= .063000
P( 5)= .032000
P( 6)= .010000
P( 7)= .004000
P( 8)= .001000
P( 9)= .001000
P(10)=0.
P(11)= .001000
P(12)=0.
P(13)=0.
P(14)=0.
P(15)=0.
P(16)=0.
P(17)=0.
P(18)=0.
P(19)=0.
P(20)=0.
```

```
TIME= 10.0000  AV.NO. IN SYS.=  1.989   STD.DEV.=  1.912

P( 0)= .269000
P( 1)= .215000
P( 2)= .189000
P( 3)= .128000
P( 4)= .087000
P( 5)= .054000
P( 6)= .029000
P( 7)= .020000
P( 8)= .004000
P( 9)= .003000
P(10)= .001000
P(11)= .001000
P(12)=0.
P(13)=0.
P(14)=0.
P(15)=0.
P(16)=0.
P(17)=0.
P(18)=0.
P(19)=0.
P(20)=0.
```

```
TIPE= 15.0000  AV.NO. IN SYS.=  2.282   STD.DEV.=  2.182

P( 0)= .233000
P( 1)= .219000
P( 2)= .179000
P( 3)= .121000
P( 4)= .093600
P( 5)= .066000
P( 6)= .040000
P( 7)= .020000
P( 8)= .015000
P( 9)= .006000
P(10)= .005000
P(11)= .002000
P(12)=0.
P(13)=0.
P(14)= .001000
P(15)=0.
P(16)=0.
P(17)=0.
P(18)=0.
P(19)=0.
P(20)=0.
```

```
 TIME= 20.0000  AV.NO. IN SYS.=  2.565   STD.DEV.=  2.506

 P( 0)= .248000
 P( 1)= .182000
 P( 2)= .150000
 P( 3)= .122000
 P( 4)= .090000
 P( 5)= .075000
 P( 6)= .056000
 P( 7)= .036000
 P( 8)= .017000
 P( 9)= .011000
 P(10)= .005000
 P(11)= .006000
 P(12)= .001000
 P(13)= .001000
 P(14)= .002000
 P(15)=0.
 P(16)=0.
 P(17)=0.
 P(18)=0.
 P(19)=0.
 P(20)=0.
```

APPENDIX B

LISTING OF BESSEL FUNCTION PROGRAM

The following is a FORTRAN listing and sample output of a program written specifically to obtain the numerical solution of the Bessel function transient analytical expression for the M/M/1 queue (15).

```
      C
      C     SOLUTION OF QUEUEING PROBLEM BY SOLVING BESSEL FUNCTIONS OF
      C     THE FIRST KIND. STUART OLSON
      C
0001        REAL LAMBDA,MU,PN(500),TITLE(20),EPS/0.00005/
0002        COMMON /$$STU/ LAMBDA,MU,ALPHA,SQRTA,SQRTB,PWRA,PWRB,TERM,EPART,
           1 COEFI,I
0003        NAMELIST /ODATA/ I,N,LAMBDA,MU,IMAX,DT
0004      8 READ (5,1,END=100) TITLE
0005      1 FORMAT (20A4)
0006        READ (5,ODATA,END=100,ERR=101)
0007        ALPHA=LAMBDA/MU
0008        PRINT 5,TITLE,I,N,LAMBDA,MU,ALPHA
0009      5 FORMAT (1H1,20A4/1H0,9X,6HINIT =,I11,4X,6HNMAX=,I9,4X,6HLAMBDA=,F13.4,2X,3HMU=,F10.5,5X,5HALPHA
           1 /1H ,2I15,3F15.4/1H0,8X,1HT,3X,8HP-VECTOR)
0010        LINE=0
0011        NSTAR=N
      C
      C     COMPUTE AUXILIARY CONSTANTS.
      C
0012        T=0.
0013        NLIM=N+1
0014        RATIO=MU/LAMBDA
0015        SQRTA=SQRT(RATIO)
0016        SQRTI=SQRTA**I
0017        SQRTB=SQRT(4.*LAMBDA*MU)
0018        GERM=LAMBDA/MU
0019        PLUS=-(LAMBDA+MU)
0020      7 PWRA=SQRTI
0021        TERM=(1.-GERM)
0022        COEFI=SQRTI*SQRTA
0023        PWRB=COEFI
0024        EPART=EXP(PLUS*T)
0025        NSTAR=0
0026        DO 3 NN=1,NLIM
0027        COEFI=COEFI*SQRTA
0028        PN(NN)=PBESSQ(NN-1,T)
0029        IF (NSTAR.EQ.0 .AND. PN(NN).LT.EPS) NSTAR=NN
0030        TERM=TERM*GERM
0031        PWRB=PWRA
0032        PWRA=PWRA/SQRTA
0033      3 CONTINUE
0034        IF (NSTAR.EQ.0) NSTAR=N
0035        LINE=LINE+(NSTAR+9)/10
0036        IF (LINE.LT.50) GO TO 10
0037        LINE=0
0038        PRINT 5,TITLE,I,N,LAMBDA,MU,ALPHA
0039     10 NNN=MIN0(10,NSTAR)
```

```
0040              PRINT 11,T,(J,PN(J),J=1,NNN)
0041           11 FORMAT (1H F9.4,10(I4,F7.4))
0042              NNN=NSTAR-NNN
0043              IF (NNN.LE.0) GO TO 13
0044              N0=NNN+1
0045              PRINT 12,(J,PN(J),J=N0,NSTAR)
0046           12 FORMAT (1H 9X10(I4,F7.4))
0047           13 T=T+DT
0048              IF (T.LE.TMAX) GO TO 7
0049              GO TO 8
0050          101 PRINT 102
0051          102 FORMAT ('0INPUT ERROR')
0052          100 CALL EXIT
0053              END
```

```
FORTRAN IV G LEVEL  20                  PBESSQ              DATE = 72103          18/05/06              PAGE 0001

0001                FUNCTION PBESSQ(N,T)
0002                REAL LAMBDA,MU
0003                COMMON /$$STU/ LAMBDA,MU,ALPHA,SQRTA,SQRTB,PWRA,PWRB,TERM,EPART,
                   1 COEFI,I
0004                IF (T.LE.0.0) GO TO 12
0005                SUM=0.
0006                X=SQRTB*T
0007                K=I+N+1
0008                KLIM=N+50
0009                COEF=COEFI
0010              3 K=K+1
0011                COEF=COEF*SQRTA
0012                CALL BESI(X,K,BI,IER)
0013                IF (IER.EQ.0) GO TO 101
0014                BI=0.
0015              101 SUM2=COEF*BI
0016                SUM=SUM+SUM2
0017                IF (.NOT.(SUM2.LT.1.E-6*SUM .OR. K.GT.KLIM)) GO TO 3
0018              2 IMN=I-N
0019                CALL BESI(X,IABS(IMN),BI,IER)
0020                IF (IER.EQ.0) GO TO 103
0021                BI=0.
0022              103 IPN=I+N+1
0023                CALL BESI(X,IPN,BIB,IER)
0024                IF (IER.EQ.0) GO TO 104
0025                BIB=0.
0026              104 PNT=EPART*(PWRA*BI+PWRB*BIB+TERM*SUM)
0027                PBESSQ=PNT
0028                RETURN
0029             12 PBESSQ=0.0
0030                IF (N.EQ.1) PBESSQ=1.0
0031                RETURN
0032                END
```

```
FORTRAN IV G LEVEL  20                  BESI              DATE = 72103          18/05/06              PAGE 0001

0001                SUBROUTINE BESI(X,N,BI,IER)
0002                DATA UNDER/1.E-50/
0003                IER=0
0004                BI=1.
0005                IF (N)150,15,10
0006             10 IF (X)160,20,20
0007             15 IF (X)160,17,20
0008             17 RETURN
0009             20 TOL=1.E-6
0010                IF (X-12.)40,40,30
0011             30 IF (X-FLOAT(N))40,40,110
0012             40 XX=X/2.
0013             50 TERM=1.
0014                IF (N)70,70,55
0015             55 DO 60 I=1,N
0016                FI=I
0017                IF (ABS(TERM)-UNDER)56,60,60
0018             56 IER=3
0019                BI=0.
0020                RETURN
0021             60 TERM=TERM*XX/FI
0022             70 BI=TERM
0023                XX=XX*XX
0024                DO 90 K=1,1000
0025                IF (ABS(TERM)-ABS(BI*TOL))100,100,80
0026             80 FK=K*(N+K)
0027                TERM=TERM*(XX/FK)
0028             90 BI=BI+TERM
0029            100 RETURN
0030            110 FN=4*N*N
0031                IF (X-170.)115,111,111
0032            111 IER=4
0033                RETURN
0034            115 XX=1./18.*X
0035                TERM=1.
0036                BI=1.
0037                DO 130 K=1,30
0038                IF (ABS(TERM)-ABS(TOL*BI))140,140,120
0039            120 FK=(2*K-1)**2
0040                TERM=TERM*XX*(FK-FN)/FLOAT(K)
0041            130 BI=BI+TERM
0042                GO TO 40
0043            140 PI=3.141593
0044                BI=BI*EXP(X)/SQRT(2.*PI*X)
0045                GO TO 100
0046            150 IER=1
0047                GO TO 100
0048            160 IER=2

FORTRAN IV G LEVEL  20                  BESI              DATE = 72103          18/05/06              PAGE 0002

0049                GO TO 100
0050                END
```

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR																			
0.0	1 1.0000	2 0.0																		
0.1000	1 0.9571	2 0.0420	3 0.0009	4 0.0000																
0.2000	1 0.9179	2 0.0785	3 0.0035	4 0.0001	5 0.0000															
0.3000	1 0.8822	2 0.1103	3 0.0072	4 0.0003	5 0.0000															
0.4000	1 0.8495	2 0.1379	3 0.0120	4 0.0007	5 0.0000															
0.5000	1 0.8195	2 0.1619	3 0.0174	4 0.0013	5 0.0001	6 0.0000														
0.6000	1 0.7919	2 0.1828	3 0.0233	4 0.0020	5 0.0001	6 0.0000														
0.7000	1 0.7665	2 0.2009	3 0.0296	4 0.0030	5 0.0002	6 0.0000														
0.8000	1 0.7431	2 0.2166	3 0.0360	4 0.0041	5 0.0004	6 0.0000														
0.9000	1 0.7214	2 0.2303	3 0.0426	4 0.0055	5 0.0005	6 0.0000														
1.0000	1 0.7012	2 0.2421	3 0.0492	4 0.0070	5 0.0008	6 0.0001	7 0.0000													
1.1000	1 0.6825	2 0.2524	3 0.0557	4 0.0087	5 0.0010	6 0.0001	7 0.0000													
1.2000	1 0.6651	2 0.2613	3 0.0622	4 0.0105	5 0.0014	6 0.0001	7 0.0000													
1.3000	1 0.6489	2 0.2689	3 0.0685	4 0.0124	5 0.0017	6 0.0002	7 0.0000													
1.4000	1 0.6337	2 0.2755	3 0.0746	4 0.0144	5 0.0022	6 0.0003	7 0.0000													
1.5000	1 0.6195	2 0.2811	3 0.0805	4 0.0166	5 0.0026	6 0.0003	7 0.0000													
1.6000	1 0.6062	2 0.2859	3 0.0862	4 0.0188	5 0.0032	6 0.0004	7 0.0001	8 0.0000												
1.7000	1 0.5936	2 0.2900	3 0.0917	4 0.0211	5 0.0038	6 0.0006	7 0.0001	8 0.0000												
1.8000	1 0.5818	2 0.2934	3 0.0970	4 0.0234	5 0.0044	6 0.0007	7 0.0001	8 0.0000												
1.9000	1 0.5707	2 0.2963	3 0.1021	4 0.0258	5 0.0051	6 0.0008	7 0.0001	8 0.0000												
2.0000	1 0.5602	2 0.2987	3 0.1069	4 0.0282	5 0.0058	6 0.0010	7 0.0001	8 0.0000												
2.1000	1 0.5503	2 0.3007	3 0.1115	4 0.0306	5 0.0066	6 0.0012	7 0.0002	8 0.0000												
2.2000	1 0.5409	2 0.3023	3 0.1158	4 0.0331	5 0.0075	6 0.0014	7 0.0002	8 0.0000												
2.3000	1 0.5319	2 0.3035	3 0.1200	4 0.0355	5 0.0083	6 0.0016	7 0.0003	8 0.0000												
2.4000	1 0.5234	2 0.3045	3 0.1239	4 0.0380	5 0.0092	6 0.0019	7 0.0003	8 0.0000												
2.5000	1 0.5154	2 0.3052	3 0.1276	4 0.0404	5 0.0102	6 0.0021	7 0.0004	8 0.0001	9 0.0000											
2.6000	1 0.5077	2 0.3057	3 0.1312	4 0.0428	5 0.0111	6 0.0024	7 0.0004	8 0.0001	9 0.0000											
2.7000	1 0.5004	2 0.3059	3 0.1345	4 0.0452	5 0.0121	6 0.0027	7 0.0005	8 0.0001	9 0.0000											
2.8000	1 0.4934	2 0.3060	3 0.1377	4 0.0475	5 0.0132	6 0.0030	7 0.0006	8 0.0001	9 0.0000											
2.9000	1 0.4867	2 0.3060	3 0.1406	4 0.0498	5 0.0142	6 0.0034	7 0.0007	8 0.0001	9 0.0000											
3.0000	1 0.4803	2 0.3058	3 0.1435	4 0.0521	5 0.0153	6 0.0037	7 0.0008	8 0.0001	9 0.0000											
3.1000	1 0.4741	2 0.3054	3 0.1461	4 0.0544	5 0.0164	6 0.0041	7 0.0009	8 0.0002	9 0.0000											
3.2000	1 0.4682	2 0.3050	3 0.1486	4 0.0566	5 0.0175	6 0.0045	7 0.0010	8 0.0002	9 0.0000											
3.3000	1 0.4626	2 0.3045	3 0.1510	4 0.0588	5 0.0186	6 0.0049	7 0.0011	8 0.0002	9 0.0000											
3.4000	1 0.4572	2 0.3039	3 0.1532	4 0.0609	5 0.0197	6 0.0054	7 0.0013	8 0.0003	9 0.0000											
3.5000	1 0.4519	2 0.3032	3 0.1553	4 0.0629	5 0.0209	6 0.0058	7 0.0014	8 0.0003	9 0.0001	10 0.0000										
3.6000	1 0.4469	2 0.3025	3 0.1573	4 0.0650	5 0.0220	6 0.0063	7 0.0015	8 0.0003	9 0.0001	10 0.0000										
3.7000	1 0.4421	2 0.3017	3 0.1592	4 0.0670	5 0.0232	6 0.0068	7 0.0017	8 0.0004	9 0.0001	10 0.0000										
3.8000	1 0.4374	2 0.3009	3 0.1609	4 0.0689	5 0.0243	6 0.0073	7 0.0019	8 0.0004	9 0.0001	10 0.0000										
3.9000	1 0.4329	2 0.3000	3 0.1626	4 0.0708	5 0.0255	6 0.0078	7 0.0020	8 0.0005	9 0.0001	10 0.0000										
4.0000	1 0.4285	2 0.2991	3 0.1641	4 0.0726	5 0.0266	6 0.0083	7 0.0022	8 0.0005	9 0.0001	10 0.0000										
4.1000	1 0.4243	2 0.2982	3 0.1656	4 0.0744	5 0.0278	6 0.0088	7 0.0024	8 0.0006	9 0.0001	10 0.0000										
4.2000	1 0.4203	2 0.2972	3 0.1669	4 0.0762	5 0.0289	6 0.0093	7 0.0026	8 0.0006	9 0.0001	10 0.0000										
4.3000	1 0.4163	2 0.2962	3 0.1682	4 0.0779	5 0.0301	6 0.0099	7 0.0028	8 0.0007	9 0.0002	10 0.0000										
4.4000	1 0.4125	2 0.2952	3 0.1694	4 0.0795	5 0.0312	6 0.0105	7 0.0030	8 0.0008	9 0.0002	10 0.0000										
4.5000	1 0.4088	2 0.2942	3 0.1705	4 0.0811	5 0.0323	6 0.0110	7 0.0033	8 0.0009	9 0.0002	10 0.0000										
4.6000	1 0.4052	2 0.2932	3 0.1716	4 0.0827	5 0.0335	6 0.0116	7 0.0035	8 0.0009	9 0.0002	10 0.0000										
4.7000	1 0.4018	2 0.2921	3 0.1726	4 0.0842	5 0.0346	6 0.0122	7 0.0037	8 0.0010	9 0.0002	10 0.0001										
	2 0.2921	3 0.1726	4 0.0842	5 0.0346	6 0.0122	7 0.0037	8 0.0010	9 0.0002	10 0.0001	11 0.0000										

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
C	21	0.4500	0.5000	0.9000

T	P-VECTOR										
4.8000	1 0.3584	2 0.2911	3 0.1735	4 0.0857	5 0.0357	6 0.0128	7 0.0040	8 0.0011	9 0.0003	10 0.0001	
	2 0.2911	3 0.1735	4 0.0857	5 0.0357	6 0.0128	7 0.0040	8 0.0011	9 0.0003	10 0.0001	11 0.0000	
4.9000	1 0.3751	2 0.2900	3 0.1744	4 0.0871	5 0.0368	6 0.0134	7 0.0042	8 0.0012	9 0.0003	10 0.0001	
	2 0.2900	3 0.1744	4 0.0871	5 0.0368	6 0.0134	7 0.0042	8 0.0012	9 0.0003	10 0.0001	11 0.0000	
5.0000	1 0.3919	2 0.2890	3 0.1752	4 0.0885	5 0.0379	6 0.0140	7 0.0045	8 0.0013	9 0.0003	10 0.0001	
	2 0.2890	3 0.1752	4 0.0885	5 0.0379	6 0.0140	7 0.0045	8 0.0013	9 0.0003	10 0.0001	11 0.0000	
5.1000	1 0.3888	2 0.2879	3 0.1760	4 0.0898	5 0.0390	6 0.0146	7 0.0048	8 0.0014	9 0.0004	10 0.0001	
	2 0.2879	3 0.1760	4 0.0898	5 0.0390	6 0.0146	7 0.0048	8 0.0014	9 0.0004	10 0.0001	11 0.0000	
5.2000	1 0.3858	2 0.2869	3 0.1767	4 0.0912	5 0.0400	6 0.0152	7 0.0050	8 0.0015	9 0.0004	10 0.0001	
	2 0.2869	3 0.1767	4 0.0912	5 0.0400	6 0.0152	7 0.0050	8 0.0015	9 0.0004	10 0.0001	11 0.0000	
5.3000	1 0.3829	2 0.2858	3 0.1773	4 0.0924	5 0.0411	6 0.0158	7 0.0053	8 0.0016	9 0.0004	10 0.0001	
	2 0.2858	3 0.1773	4 0.0924	5 0.0411	6 0.0158	7 0.0053	8 0.0016	9 0.0004	10 0.0001	11 0.0000	
5.4000	1 0.3800	2 0.2847	3 0.1779	4 0.0937	5 0.0421	6 0.0164	7 0.0056	8 0.0017	9 0.0005	10 0.0001	
	2 0.2847	3 0.1779	4 0.0937	5 0.0421	6 0.0164	7 0.0056	8 0.0017	9 0.0005	10 0.0001	11 0.0000	
5.5000	1 0.3772	2 0.2837	3 0.1785	4 0.0949	5 0.0431	6 0.0170	7 0.0059	8 0.0018	9 0.0005	10 0.0001	
	2 0.2837	3 0.1785	4 0.0949	5 0.0431	6 0.0170	7 0.0059	8 0.0018	9 0.0005	10 0.0001	11 0.0000	
5.6000	1 0.3745	2 0.2826	3 0.1790	4 0.0960	5 0.0442	6 0.0176	7 0.0062	8 0.0019	9 0.0005	10 0.0001	
	2 0.2826	3 0.1790	4 0.0960	5 0.0442	6 0.0176	7 0.0062	8 0.0019	9 0.0005	10 0.0001	11 0.0000	
5.7000	1 0.3719	2 0.2816	3 0.1795	4 0.0971	5 0.0452	6 0.0183	7 0.0065	8 0.0021	9 0.0006	10 0.0002	
	2 0.2816	3 0.1795	4 0.0971	5 0.0452	6 0.0183	7 0.0065	8 0.0021	9 0.0006	10 0.0002	11 0.0000	
5.8000	1 0.3693	2 0.2806	3 0.1800	4 0.0982	5 0.0461	6 0.0189	7 0.0068	8 0.0022	9 0.0006	10 0.0002	
	2 0.2806	3 0.1800	4 0.0982	5 0.0461	6 0.0189	7 0.0068	8 0.0022	9 0.0006	10 0.0002	11 0.0000	
5.9000	1 0.3668	2 0.2795	3 0.1804	4 0.0993	5 0.0471	6 0.0195	7 0.0071	8 0.0023	9 0.0007	10 0.0002	
	2 0.2795	3 0.1804	4 0.0993	5 0.0471	6 0.0195	7 0.0071	8 0.0023	9 0.0007	10 0.0002	11 0.0000	
6.0000	1 0.3643	2 0.2785	3 0.1808	4 0.1003	5 0.0481	6 0.0201	7 0.0075	8 0.0025	9 0.0007	10 0.0002	
	2 0.2785	3 0.1808	4 0.1003	5 0.0481	6 0.0201	7 0.0075	8 0.0025	9 0.0007	10 0.0002	11 0.0000	
6.1000	1 0.3619	2 0.2775	3 0.1812	4 0.1013	5 0.0490	6 0.0208	7 0.0078	8 0.0026	9 0.0008	10 0.0002	
	3 0.1812	4 0.1013	5 0.0490	6 0.0208	7 0.0078	8 0.0026	9 0.0008	10 0.0002	11 0.0001	12 0.0000	
6.2000	1 0.3596	2 0.2765	3 0.1815	4 0.1023	5 0.0500	6 0.0214	7 0.0081	8 0.0027	9 0.0008	10 0.0002	
	3 0.1815	4 0.1023	5 0.0500	6 0.0214	7 0.0081	8 0.0027	9 0.0008	10 0.0002	11 0.0001	12 0.0000	
6.3000	1 0.3573	2 0.2755	3 0.1818	4 0.1032	5 0.0509	6 0.0220	7 0.0084	8 0.0029	9 0.0009	10 0.0003	
	3 0.1818	4 0.1032	5 0.0509	6 0.0220	7 0.0084	8 0.0029	9 0.0009	10 0.0003	11 0.0001	12 0.0000	
6.4000	1 0.3551	2 0.2745	3 0.1821	4 0.1041	5 0.0518	6 0.0226	7 0.0088	8 0.0030	9 0.0010	10 0.0003	
	3 0.1821	4 0.1041	5 0.0518	6 0.0226	7 0.0088	8 0.0030	9 0.0010	10 0.0003	11 0.0001	12 0.0000	
6.5000	1 0.3529	2 0.2735	3 0.1823	4 0.1050	5 0.0527	6 0.0232	7 0.0091	8 0.0032	9 0.0010	10 0.0003	
	3 0.1823	4 0.1050	5 0.0527	6 0.0232	7 0.0091	8 0.0032	9 0.0010	10 0.0003	11 0.0001	12 0.0000	
6.6000	1 0.3507	2 0.2725	3 0.1825	4 0.1058	5 0.0535	6 0.0239	7 0.0095	8 0.0034	9 0.0011	10 0.0003	
	3 0.1825	4 0.1058	5 0.0535	6 0.0239	7 0.0095	8 0.0034	9 0.0011	10 0.0003	11 0.0001	12 0.0000	
6.7000	1 0.3487	2 0.2715	3 0.1827	4 0.1067	5 0.0544	6 0.0245	7 0.0098	8 0.0035	9 0.0011	10 0.0003	
	3 0.1827	4 0.1067	5 0.0544	6 0.0245	7 0.0098	8 0.0035	9 0.0011	10 0.0003	11 0.0001	12 0.0000	
6.8000	1 0.3466	2 0.2706	3 0.1829	4 0.1075	5 0.0553	6 0.0251	7 0.0101	8 0.0037	9 0.0012	10 0.0004	
	3 0.1829	4 0.1075	5 0.0553	6 0.0251	7 0.0101	8 0.0037	9 0.0012	10 0.0004	11 0.0001	12 0.0000	
6.9000	1 0.3446	2 0.2696	3 0.1831	4 0.1082	5 0.0561	6 0.0257	7 0.0105	8 0.0039	9 0.0013	10 0.0004	
	3 0.1831	4 0.1082	5 0.0561	6 0.0257	7 0.0105	8 0.0039	9 0.0013	10 0.0004	11 0.0001	12 0.0000	
7.0000	1 0.3426	2 0.2687	3 0.1832	4 0.1090	5 0.0569	6 0.0263	7 0.0108	8 0.0040	9 0.0014	10 0.0004	
	3 0.1832	4 0.1090	5 0.0569	6 0.0263	7 0.0108	8 0.0040	9 0.0014	10 0.0004	11 0.0001	12 0.0000	
7.1000	1 0.3406	2 0.2677	3 0.1833	4 0.1097	5 0.0577	6 0.0269	7 0.0112	8 0.0042	9 0.0014	10 0.0004	
	3 0.1833	4 0.1097	5 0.0577	6 0.0269	7 0.0112	8 0.0042	9 0.0014	10 0.0004	11 0.0001	12 0.0000	
7.2000	1 0.3386	2 0.2668	3 0.1833	4 0.1104	5 0.0585	6 0.0275	7 0.0116	8 0.0044	9 0.0015	10 0.0005	
	3 0.1833	4 0.1104	5 0.0585	6 0.0275	7 0.0116	8 0.0044	9 0.0015	10 0.0005	11 0.0001	12 0.0000	

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
C	21	0.4500	0.5000	0.9000

T	P-VECTOR											
7.3000	1 0.3370	2 0.2659	3 0.1835	4 0.1111	5 0.0593	6 0.0281	7 0.0119	8 0.0046	9 0.0016	10 0.0005		
	3 0.1835	4 0.1111	5 0.0593	6 0.0281	7 0.0119	8 0.0046	9 0.0016	10 0.0005	11 0.0001	12 0.0000		
7.4000	1 0.3352	2 0.2650	3 0.1836	4 0.1118	5 0.0601	6 0.0287	7 0.0123	8 0.0047	9 0.0017	10 0.0005		
	3 0.1836	4 0.1118	5 0.0601	6 0.0287	7 0.0123	8 0.0047	9 0.0017	10 0.0005	11 0.0002	12 0.0000		
7.5000	1 0.3334	2 0.2641	3 0.1837	4 0.1124	5 0.0608	6 0.0293	7 0.0126	8 0.0049	9 0.0017	10 0.0006		
	3 0.1837	4 0.1124	5 0.0608	6 0.0293	7 0.0126	8 0.0049	9 0.0017	10 0.0006	11 0.0002	12 0.0000		
7.6000	1 0.3316	2 0.2632	3 0.1837	4 0.1130	5 0.0615	6 0.0299	7 0.0130	8 0.0051	9 0.0018	10 0.0006		
	4 0.1130	5 0.0615	6 0.0299	7 0.0130	8 0.0051	9 0.0018	10 0.0006	11 0.0002	12 0.0001	13 0.0000		
7.7000	1 0.3299	2 0.2623	3 0.1838	4 0.1136	5 0.0623	6 0.0304	7 0.0134	8 0.0053	9 0.0019	10 0.0006		
	4 0.1136	5 0.0623	6 0.0304	7 0.0134	8 0.0053	9 0.0019	10 0.0006	11 0.0002	12 0.0001	13 0.0000		
7.8000	1 0.3283	2 0.2614	3 0.1838	4 0.1142	5 0.0630	6 0.0310	7 0.0137	8 0.0055	9 0.0020	10 0.0007		
	4 0.1142	5 0.0630	6 0.0310	7 0.0137	8 0.0055	9 0.0020	10 0.0007	11 0.0002	12 0.0001	13 0.0000		
7.9000	1 0.3266	2 0.2605	3 0.1838	4 0.1148	5 0.0637	6 0.0316	7 0.0141	8 0.0057	9 0.0021	10 0.0007		
	4 0.1148	5 0.0637	6 0.0316	7 0.0141	8 0.0057	9 0.0021	10 0.0007	11 0.0002	12 0.0001	13 0.0000		
8.0000	1 0.3250	2 0.2597	3 0.1838	4 0.1153	5 0.0644	6 0.0322	7 0.0145	8 0.0059	9 0.0022	10 0.0007		
	4 0.1153	5 0.0644	6 0.0322	7 0.0145	8 0.0059	9 0.0022	10 0.0007	11 0.0002	12 0.0001	13 0.0000		
8.1000	1 0.3234	2 0.2588	3 0.1838	4 0.1158	5 0.0650	6 0.0327	7 0.0148	8 0.0061	9 0.0023	10 0.0008		
	4 0.1158	5 0.0650	6 0.0327	7 0.0148	8 0.0061	9 0.0023	10 0.0008	11 0.0003	12 0.0001	13 0.0000		
8.2000	1 0.3218	2 0.2580	3 0.1837	4 0.1163	5 0.0657	6 0.0333	7 0.0152	8 0.0063	9 0.0024	10 0.0008		
	4 0.1163	5 0.0657	6 0.0333	7 0.0152	8 0.0063	9 0.0024	10 0.0008	11 0.0003	12 0.0001	13 0.0000		
8.3000	1 0.3203	2 0.2571	3 0.1837	4 0.1168	5 0.0664	6 0.0338	7 0.0156	8 0.0065	9 0.0025	10 0.0009		
	4 0.1168	5 0.0664	6 0.0338	7 0.0156	8 0.0065	9 0.0025	10 0.0009	11 0.0003	12 0.0001	13 0.0000		
8.4000	1 0.3188	2 0.2563	3 0.1837	4 0.1173	5 0.0670	6 0.0344	7 0.0159	8 0.0067	9 0.0026	10 0.0009		
	4 0.1173	5 0.0670	6 0.0344	7 0.0159	8 0.0067	9 0.0026	10 0.0009	11 0.0003	12 0.0001	13 0.0000		
8.5000	1 0.3173	2 0.2555	3 0.1836	4 0.1178	5 0.0676	6 0.0349	7 0.0163	8 0.0069	9 0.0027	10 0.0010		
	4 0.1178	5 0.0676	6 0.0349	7 0.0163	8 0.0069	9 0.0027	10 0.0010	11 0.0003	12 0.0001	13 0.0000		
8.6000	1 0.3159	2 0.2547	3 0.1835	4 0.1182	5 0.0682	6 0.0355	7 0.0167	8 0.0071	9 0.0028	10 0.0010		
	4 0.1182	5 0.0682	6 0.0355	7 0.0167	8 0.0071	9 0.0028	10 0.0010	11 0.0003	12 0.0001	13 0.0000		
8.7000	1 0.3144	2 0.2539	3 0.1835	4 0.1186	5 0.0688	6 0.0360	7 0.0170	8 0.0074	9 0.0029	10 0.0011		
	4 0.1186	5 0.0688	6 0.0360	7 0.0170	8 0.0074	9 0.0029	10 0.0011	11 0.0004	12 0.0001	13 0.0000		
8.8000	1 0.3130	2 0.2531	3 0.1834	4 0.1191	5 0.0694	6 0.0365	7 0.0174	8 0.0076	9 0.0030	10 0.0011		
	4 0.1191	5 0.0694	6 0.0365	7 0.0174	8 0.0076	9 0.0030	10 0.0011	11 0.0004	12 0.0001	13 0.0000		
8.9000	1 0.3117	2 0.2523	3 0.1833	4 0.1195	5 0.0700	6 0.0370	7 0.0178	8 0.0078	9 0.0031	10 0.0012		
	4 0.1195	5 0.0700	6 0.0370	7 0.0178	8 0.0078	9 0.0031	10 0.0012	11 0.0004	12 0.0001	13 0.0000		
9.0000	1 0.3103	2 0.2515	3 0.1832	4 0.1199	5 0.0706	6 0.0376	7 0.0182	8 0.0080	9 0.0032	10 0.0012		
	4 0.1199	5 0.0706	6 0.0376	7 0.0182	8 0.0080	9 0.0032	10 0.0012	11 0.0004	12 0.0001	13 0.0000		
9.1000	1 0.3090	2 0.2508	3 0.1831	4 0.1203	5 0.0712	6 0.0381	7 0.0185	8 0.0082	9 0.0033	10 0.0013		
	4 0.1203	5 0.0712	6 0.0381	7 0.0185	8 0.0082	9 0.0033	10 0.0013	11 0.0004	12 0.0001	13 0.0000		
9.2000	1 0.3076	2 0.2501	3 0.1829	4 0.1207	5 0.0717	6 0.0386	7 0.0189	8 0.0084	9 0.0035	10 0.0013		
	4 0.1207	5 0.0717	6 0.0386	7 0.0189	8 0.0084	9 0.0035	10 0.0013	11 0.0005	12 0.0002	13 0.0000		
9.3000	1 0.3063	2 0.2494	3 0.1827	4 0.1210	5 0.0722	6 0.0391	7 0.0192	8 0.0087	9 0.0036	10 0.0014		
	4 0.1210	5 0.0722	6 0.0391	7 0.0192	8 0.0087	9 0.0036	10 0.0014	11 0.0005	12 0.0002	13 0.0000		
9.4000	1 0.3051	2 0.2488	3 0.1824	4 0.1213	5 0.0728	6 0.0396	7 0.0196	8 0.0089	9 0.0037	10 0.0014		
	5 0.0728	6 0.0396	7 0.0196	8 0.0089	9 0.0037	10 0.0014	11 0.0005	12 0.0002	13 0.0001	14 0.0000		
9.5000	1 0.3039	2 0.2481	3 0.1821	4 0.1217	5 0.0733	6 0.0401	7 0.0200	8 0.0091	9 0.0038	10 0.0015		
	5 0.0733	6 0.0401	7 0.0200	8 0.0091	9 0.0038	10 0.0015	11 0.0005	12 0.0002	13 0.0001	14 0.0000		
9.6000	1 0.3028	2 0.2474	3 0.1818	4 0.1220	5 0.0738	6 0.0406	7 0.0203	8 0.0093	9 0.0039	10 0.0015		
	5 0.0738	6 0.0406	7 0.0203	8 0.0093	9 0.0039	10 0.0015	11 0.0006	12 0.0002	13 0.0001	14 0.0000		
9.7000	1 0.3016	2 0.2468	3 0.1814	4 0.1223	5 0.0743	6 0.0410	7 0.0207	8 0.0096	9 0.0041	10 0.0016		
	5 0.0743	6 0.0410	7 0.0207	8 0.0096	9 0.0041	10 0.0016	11 0.0006	12 0.0002	13 0.0001	14 0.0000		

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR													
9.7999	1 0.3091	2 0.2456	3 0.1823	4 0.1226	5 0.0748	6 0.0415	7 0.0211	8 0.0098	9 0.0042	10 0.0017				
	5 0.0748	6 0.0415	7 0.0211	8 0.0098	9 0.0042	10 0.0017	11 0.0006	12 0.0002	13 0.0001	14 0.0000				
9.8999	1 0.2990	2 0.2449	3 0.1822	4 0.1229	5 0.0753	6 0.0420	7 0.0214	8 0.0100	9 0.0043	10 0.0017				
	5 0.0753	6 0.0420	7 0.0214	8 0.0100	9 0.0043	10 0.0017	11 0.0006	12 0.0002	13 0.0001	14 0.0000				
9.9999	1 0.2978	2 0.2442	3 0.1820	4 0.1232	5 0.0758	6 0.0425	7 0.0218	8 0.0102	9 0.0044	10 0.0018				
	5 0.0758	6 0.0425	7 0.0218	8 0.0102	9 0.0044	10 0.0018	11 0.0007	12 0.0002	13 0.0001	14 0.0000				
10.0999	1 0.2967	2 0.2435	3 0.1819	4 0.1234	5 0.0762	6 0.0429	7 0.0221	8 0.0105	9 0.0046	10 0.0018				
	5 0.0762	6 0.0429	7 0.0221	8 0.0105	9 0.0046	10 0.0018	11 0.0007	12 0.0002	13 0.0001	14 0.0000				
10.1999	1 0.2955	2 0.2428	3 0.1817	4 0.1237	5 0.0767	6 0.0434	7 0.0225	8 0.0107	9 0.0047	10 0.0019				
	5 0.0767	6 0.0434	7 0.0225	8 0.0107	9 0.0047	10 0.0019	11 0.0007	12 0.0003	13 0.0001	14 0.0000				
10.2999	1 0.2944	2 0.2421	3 0.1816	4 0.1240	5 0.0771	6 0.0438	7 0.0228	8 0.0109	9 0.0048	10 0.0020				
	5 0.0771	6 0.0438	7 0.0228	8 0.0109	9 0.0048	10 0.0020	11 0.0008	12 0.0003	13 0.0001	14 0.0000				
10.3999	1 0.2933	2 0.2415	3 0.1814	4 0.1242	5 0.0776	6 0.0443	7 0.0232	8 0.0112	9 0.0050	10 0.0020				
	5 0.0776	6 0.0443	7 0.0232	8 0.0112	9 0.0050	10 0.0020	11 0.0008	12 0.0003	13 0.0001	14 0.0000				
10.4999	1 0.2922	2 0.2408	3 0.1813	4 0.1244	5 0.0780	6 0.0447	7 0.0235	8 0.0114	9 0.0051	10 0.0021				
	5 0.0780	6 0.0447	7 0.0235	8 0.0114	9 0.0051	10 0.0021	11 0.0008	12 0.0003	13 0.0001	14 0.0000				
10.5999	1 0.2912	2 0.2401	3 0.1811	4 0.1247	5 0.0784	6 0.0451	7 0.0239	8 0.0116	9 0.0052	10 0.0022				
	5 0.0784	6 0.0451	7 0.0239	8 0.0116	9 0.0052	10 0.0022	11 0.0009	12 0.0003	13 0.0001	14 0.0000				
10.6999	1 0.2901	2 0.2395	3 0.1809	4 0.1249	5 0.0788	6 0.0456	7 0.0242	8 0.0119	9 0.0054	10 0.0023				
	5 0.0788	6 0.0456	7 0.0242	8 0.0119	9 0.0054	10 0.0023	11 0.0009	12 0.0003	13 0.0001	14 0.0000				
10.7999	1 0.2891	2 0.2388	3 0.1808	4 0.1251	5 0.0792	6 0.0460	7 0.0246	8 0.0121	9 0.0055	10 0.0023				
	5 0.0792	6 0.0460	7 0.0246	8 0.0121	9 0.0055	10 0.0023	11 0.0009	12 0.0003	13 0.0001	14 0.0000				
10.8999	1 0.2880	2 0.2382	3 0.1806	4 0.1253	5 0.0796	6 0.0464	7 0.0249	8 0.0123	9 0.0056	10 0.0024				
	5 0.0796	6 0.0464	7 0.0249	8 0.0123	9 0.0056	10 0.0024	11 0.0010	12 0.0004	13 0.0001	14 0.0000				
10.9999	1 0.2870	2 0.2376	3 0.1804	4 0.1255	5 0.0800	6 0.0468	7 0.0252	8 0.0126	9 0.0059	10 0.0025				
	5 0.0800	6 0.0468	7 0.0252	8 0.0126	9 0.0058	10 0.0025	11 0.0010	12 0.0004	13 0.0001	14 0.0000				
11.0999	1 0.2860	2 0.2369	3 0.1802	4 0.1257	5 0.0804	6 0.0473	7 0.0256	8 0.0128	9 0.0059	10 0.0026				
	5 0.0804	6 0.0473	7 0.0256	8 0.0128	9 0.0059	10 0.0026	11 0.0010	12 0.0004	13 0.0001	14 0.0000				
11.1999	1 0.2850	2 0.2363	3 0.1801	4 0.1259	5 0.0808	6 0.0477	7 0.0259	8 0.0130	9 0.0061	10 0.0026				
	5 0.0808	6 0.0477	7 0.0259	8 0.0130	9 0.0061	10 0.0026	11 0.0011	12 0.0004	13 0.0001	14 0.0000				
11.2999	1 0.2841	2 0.2357	3 0.1799	4 0.1261	5 0.0812	6 0.0481	7 0.0262	8 0.0132	9 0.0062	10 0.0027				
	5 0.0812	6 0.0481	7 0.0262	8 0.0132	9 0.0062	10 0.0027	11 0.0011	12 0.0004	13 0.0001	14 0.0001	15 0.0000			
11.3999	1 0.2831	2 0.2351	3 0.1797	4 0.1262	5 0.0815	6 0.0485	7 0.0266	8 0.0135	9 0.0063	10 0.0028				
	5 0.0815	6 0.0485	7 0.0266	8 0.0135	9 0.0063	10 0.0028	11 0.0011	12 0.0004	13 0.0002	14 0.0001	15 0.0000			
11.4999	1 0.2822	2 0.2345	3 0.1795	4 0.1264	5 0.0819	6 0.0488	7 0.0269	8 0.0137	9 0.0065	10 0.0029				
	5 0.0819	6 0.0488	7 0.0269	8 0.0137	9 0.0065	10 0.0029	11 0.0012	12 0.0005	13 0.0002	14 0.0001	15 0.0000			
11.5999	1 0.2812	2 0.2339	3 0.1793	4 0.1266	5 0.0822	6 0.0492	7 0.0272	8 0.0139	9 0.0066	10 0.0029				
	5 0.0822	6 0.0492	7 0.0272	8 0.0139	9 0.0066	10 0.0029	11 0.0012	12 0.0005	13 0.0002	14 0.0001	15 0.0000			
11.6999	1 0.2803	2 0.2333	3 0.1792	4 0.1267	5 0.0826	6 0.0496	7 0.0276	8 0.0142	9 0.0068	10 0.0030				
	5 0.0826	6 0.0496	7 0.0276	8 0.0142	9 0.0068	10 0.0030	11 0.0013	12 0.0005	13 0.0002	14 0.0001	15 0.0000			
11.7999	1 0.2794	2 0.2327	3 0.1790	4 0.1269	5 0.0829	6 0.0500	7 0.0279	8 0.0144	9 0.0069	10 0.0031				
	5 0.0829	6 0.0500	7 0.0279	8 0.0144	9 0.0069	10 0.0031	11 0.0013	12 0.0005	13 0.0002	14 0.0001	15 0.0000			
11.8999	1 0.2785	2 0.2321	3 0.1788	4 0.1270	5 0.0832	6 0.0504	7 0.0282	8 0.0146	9 0.0071	10 0.0032				
	5 0.0832	6 0.0504	7 0.0282	8 0.0146	9 0.0071	10 0.0032	11 0.0013	12 0.0005	13 0.0002	14 0.0001	15 0.0000			
11.9999	1 0.2776	2 0.2315	3 0.1786	4 0.1272	5 0.0836	6 0.0507	7 0.0285	8 0.0149	9 0.0072	10 0.0033				
	5 0.0836	6 0.0507	7 0.0285	8 0.0149	9 0.0072	10 0.0033	11 0.0014	12 0.0006	13 0.0002	14 0.0001	15 0.0000			
12.0999	1 0.2767	2 0.2310	3 0.1784	4 0.1273	5 0.0839	6 0.0511	7 0.0288	8 0.0151	9 0.0074	10 0.0034				
	5 0.0839	6 0.0511	7 0.0288	8 0.0151	9 0.0074	10 0.0034	11 0.0014	12 0.0006	13 0.0002	14 0.0001	15 0.0000			
12.1999	1 0.2759	2 0.2304	3 0.1782	4 0.1274	5 0.0842	6 0.0515	7 0.0291	8 0.0153	9 0.0075	10 0.0034				
	5 0.0842	6 0.0515	7 0.0291	8 0.0153	9 0.0075	10 0.0034	11 0.0015	12 0.0006	13 0.0002	14 0.0001	15 0.0000			

M/K/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR									
12.2999	1 0.2730	2 0.2208	3 0.1780	4 0.1275	5 0.0845	6 0.0518	7 0.0295	8 0.0156	9 0.0077	10 0.0035
	6 0.0518	7 0.0295	8 0.0156	9 0.0077	10 0.0035	11 0.0015	12 0.0006	13 0.0002	14 0.0001	15 0.0000
12.3999	1 0.2742	2 0.2213	3 0.1778	4 0.1277	5 0.0848	6 0.0522	7 0.0298	8 0.0158	9 0.0078	10 0.0036
	6 0.0522	7 0.0298	8 0.0158	9 0.0078	10 0.0036	11 0.0016	12 0.0006	13 0.0002	14 0.0001	15 0.0000
12.4999	1 0.2733	2 0.2207	3 0.1776	4 0.1278	5 0.0851	6 0.0525	7 0.0301	8 0.0160	9 0.0080	10 0.0037
	6 0.0525	7 0.0301	8 0.0160	9 0.0080	10 0.0037	11 0.0016	12 0.0007	13 0.0003	14 0.0001	15 0.0000
12.5999	1 0.2725	2 0.2202	3 0.1774	4 0.1279	5 0.0854	6 0.0529	7 0.0304	8 0.0163	9 0.0081	10 0.0038
	6 0.0529	7 0.0304	8 0.0163	9 0.0081	10 0.0038	11 0.0017	12 0.0007	13 0.0003	14 0.0001	15 0.0000
12.6999	1 0.2717	2 0.2196	3 0.1772	4 0.1280	5 0.0857	6 0.0532	7 0.0307	8 0.0165	9 0.0083	10 0.0039
	6 0.0532	7 0.0307	8 0.0165	9 0.0083	10 0.0039	11 0.0017	12 0.0007	13 0.0003	14 0.0001	15 0.0000
12.7999	1 0.2709	2 0.2191	3 0.1771	4 0.1281	5 0.0859	6 0.0535	7 0.0310	8 0.0167	9 0.0084	10 0.0040
	6 0.0535	7 0.0310	8 0.0167	9 0.0084	10 0.0040	11 0.0018	12 0.0007	13 0.0003	14 0.0001	15 0.0000
12.8999	1 0.2701	2 0.2186	3 0.1769	4 0.1282	5 0.0862	6 0.0539	7 0.0313	8 0.0169	9 0.0086	10 0.0041
	6 0.0539	7 0.0313	8 0.0169	9 0.0086	10 0.0041	11 0.0018	12 0.0008	13 0.0003	14 0.0001	15 0.0000
12.9999	1 0.2693	2 0.2181	3 0.1767	4 0.1283	5 0.0865	6 0.0542	7 0.0316	8 0.0172	9 0.0087	10 0.0041
	6 0.0542	7 0.0316	8 0.0172	9 0.0087	10 0.0041	11 0.0018	12 0.0008	13 0.0003	14 0.0001	15 0.0000
13.0999	1 0.2685	2 0.2175	3 0.1765	4 0.1283	5 0.0867	6 0.0545	7 0.0319	8 0.0174	9 0.0089	10 0.0042
	6 0.0545	7 0.0319	8 0.0174	9 0.0089	10 0.0042	11 0.0019	12 0.0008	13 0.0003	14 0.0001	15 0.0000
13.1999	1 0.2677	2 0.2170	3 0.1763	4 0.1284	5 0.0870	6 0.0548	7 0.0322	8 0.0176	9 0.0090	10 0.0043
	6 0.0548	7 0.0322	8 0.0176	9 0.0090	10 0.0043	11 0.0019	12 0.0008	13 0.0003	14 0.0001	15 0.0000
13.2999	1 0.2670	2 0.2165	3 0.1761	4 0.1285	5 0.0873	6 0.0551	7 0.0325	8 0.0178	9 0.0092	10 0.0044
	6 0.0551	7 0.0325	8 0.0178	9 0.0092	10 0.0044	11 0.0020	12 0.0009	13 0.0003	14 0.0001	15 0.0000
13.3999	1 0.2662	2 0.2160	3 0.1759	4 0.1286	5 0.0875	6 0.0554	7 0.0327	8 0.0181	9 0.0093	10 0.0045
	6 0.0554	7 0.0327	8 0.0181	9 0.0093	10 0.0045	11 0.0021	12 0.0009	13 0.0004	14 0.0001	15 0.0000
13.4999	1 0.2655	2 0.2155	3 0.1757	4 0.1287	5 0.0877	6 0.0557	7 0.0330	8 0.0183	9 0.0095	10 0.0046
	7 0.0330	8 0.0183	9 0.0095	10 0.0046	11 0.0021	12 0.0009	13 0.0004	14 0.0001	15 0.0001	16 0.0000
13.5999	1 0.2647	2 0.2150	3 0.1755	4 0.1287	5 0.0880	6 0.0560	7 0.0333	8 0.0185	9 0.0096	10 0.0047
	7 0.0333	8 0.0185	9 0.0096	10 0.0047	11 0.0022	12 0.0009	13 0.0004	14 0.0001	15 0.0001	16 0.0000
13.6999	1 0.2640	2 0.2145	3 0.1753	4 0.1288	5 0.0882	6 0.0563	7 0.0336	8 0.0187	9 0.0097	10 0.0048
	7 0.0336	8 0.0187	9 0.0097	10 0.0048	11 0.0022	12 0.0010	13 0.0004	14 0.0002	15 0.0001	16 0.0000
13.7999	1 0.2633	2 0.2140	3 0.1751	4 0.1289	5 0.0884	6 0.0566	7 0.0339	8 0.0190	9 0.0099	10 0.0049
	7 0.0339	8 0.0190	9 0.0099	10 0.0049	11 0.0023	12 0.0010	13 0.0004	14 0.0002	15 0.0001	16 0.0000
13.8999	1 0.2626	2 0.2135	3 0.1749	4 0.1289	5 0.0887	6 0.0569	7 0.0342	8 0.0192	9 0.0101	10 0.0050
	7 0.0342	8 0.0192	9 0.0101	10 0.0050	11 0.0023	12 0.0010	13 0.0004	14 0.0002	15 0.0001	16 0.0000
13.9999	1 0.2619	2 0.2130	3 0.1747	4 0.1290	5 0.0889	6 0.0572	7 0.0344	8 0.0194	9 0.0102	10 0.0051
	7 0.0344	8 0.0194	9 0.0102	10 0.0051	11 0.0024	12 0.0010	13 0.0004	14 0.0002	15 0.0001	16 0.0000
14.0999	1 0.2612	2 0.2125	3 0.1745	4 0.1290	5 0.0891	6 0.0575	7 0.0347	8 0.0196	9 0.0104	10 0.0052
	7 0.0347	8 0.0196	9 0.0104	10 0.0052	11 0.0024	12 0.0011	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.1999	1 0.2605	2 0.2121	3 0.1743	4 0.1291	5 0.0893	6 0.0578	7 0.0350	8 0.0198	9 0.0105	10 0.0053
	7 0.0350	8 0.0198	9 0.0105	10 0.0053	11 0.0025	12 0.0011	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.2999	1 0.2598	2 0.2116	3 0.1741	4 0.1291	5 0.0895	6 0.0581	7 0.0352	8 0.0200	9 0.0107	10 0.0054
	7 0.0352	8 0.0200	9 0.0107	10 0.0054	11 0.0025	12 0.0011	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.3999	1 0.2591	2 0.2111	3 0.1739	4 0.1292	5 0.0897	6 0.0583	7 0.0355	8 0.0203	9 0.0109	10 0.0055
	7 0.0355	8 0.0203	9 0.0109	10 0.0055	11 0.0026	12 0.0012	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.4999	1 0.2584	2 0.2107	3 0.1737	4 0.1292	5 0.0899	6 0.0586	7 0.0358	8 0.0205	9 0.0110	10 0.0056
	7 0.0358	8 0.0205	9 0.0110	10 0.0056	11 0.0027	12 0.0012	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.5999	1 0.2578	2 0.2102	3 0.1735	4 0.1292	5 0.0901	6 0.0589	7 0.0360	8 0.0207	9 0.0112	10 0.0057
	7 0.0360	8 0.0207	9 0.0112	10 0.0057	11 0.0027	12 0.0012	13 0.0005	14 0.0002	15 0.0001	16 0.0000
14.6999	1 0.2571	2 0.2178	3 0.1733	4 0.1293	5 0.0903	6 0.0591	7 0.0363	8 0.0209	9 0.0113	10 0.0058
	7 0.0363	8 0.0209	9 0.0113	10 0.0058	11 0.0028	12 0.0013	13 0.0005	14 0.0002	15 0.0001	16 0.0000

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR															
14.7999	1 0.2565	2 0.2173	3 0.1731	4 0.1293	5 0.0905	6 0.0594	7 0.0366	8 0.0211	9 0.0115	10 0.0059						
	7 0.0366	8 0.0211	9 0.0115	10 0.0059	11 0.0028	12 0.0013	13 0.0006	14 0.0002	15 0.0001	16 0.0000						
14.8999	1 0.2558	2 0.2169	3 0.1729	4 0.1293	5 0.0907	6 0.0597	7 0.0368	8 0.0213	9 0.0116	10 0.0060						
	7 0.0368	8 0.0213	9 0.0116	10 0.0060	11 0.0029	12 0.0013	13 0.0006	14 0.0002	15 0.0001	16 0.0000						
14.9999	1 0.2552	2 0.2164	3 0.1727	4 0.1294	5 0.0909	6 0.0599	7 0.0371	8 0.0215	9 0.0118	10 0.0061						
	7 0.0371	8 0.0215	9 0.0118	10 0.0061	11 0.0030	12 0.0014	13 0.0006	14 0.0002	15 0.0001	16 0.0000						
15.0999	1 0.2546	2 0.2160	3 0.1725	4 0.1294	5 0.0911	6 0.0602	7 0.0373	8 0.0218	9 0.0119	10 0.0062						
	7 0.0373	8 0.0218	9 0.0119	10 0.0062	11 0.0030	12 0.0014	13 0.0006	14 0.0003	15 0.0001	16 0.0000						
15.1999	1 0.2539	2 0.2155	3 0.1723	4 0.1294	5 0.0913	6 0.0604	7 0.0376	8 0.0220	9 0.0121	10 0.0063						
	7 0.0376	8 0.0220	9 0.0121	10 0.0063	11 0.0031	12 0.0014	13 0.0006	14 0.0003	15 0.0001	16 0.0000						
15.2999	1 0.2533	2 0.2151	3 0.1721	4 0.1294	5 0.0914	6 0.0606	7 0.0378	8 0.0222	9 0.0122	10 0.0064						
	7 0.0378	8 0.0222	9 0.0122	10 0.0064	11 0.0031	12 0.0015	13 0.0006	14 0.0003	15 0.0001	16 0.0000						
15.3999	1 0.2527	2 0.2147	3 0.1719	4 0.1294	5 0.0916	6 0.0609	7 0.0381	8 0.0224	9 0.0124	10 0.0065						
	7 0.0381	8 0.0224	9 0.0124	10 0.0065	11 0.0032	12 0.0015	13 0.0007	14 0.0003	15 0.0001	16 0.0000						
15.4999	1 0.2521	2 0.2143	3 0.1717	4 0.1295	5 0.0918	6 0.0611	7 0.0383	8 0.0226	9 0.0125	10 0.0066						
	7 0.0383	8 0.0226	9 0.0125	10 0.0066	11 0.0033	12 0.0015	13 0.0007	14 0.0003	15 0.0001	16 0.0000						
15.5999	1 0.2515	2 0.2138	3 0.1715	4 0.1295	5 0.0919	6 0.0614	7 0.0385	8 0.0228	9 0.0127	10 0.0067						
	7 0.0385	8 0.0228	9 0.0127	10 0.0067	11 0.0033	12 0.0016	13 0.0007	14 0.0003	15 0.0001	16 0.0000						
15.6999	1 0.2509	2 0.2134	3 0.1713	4 0.1295	5 0.0921	6 0.0616	7 0.0388	8 0.0230	9 0.0129	10 0.0068						
	7 0.0388	8 0.0230	9 0.0129	10 0.0068	11 0.0034	12 0.0016	13 0.0007	14 0.0003	15 0.0001	16 0.0000						
15.7999	1 0.2503	2 0.2130	3 0.1711	4 0.1295	5 0.0922	6 0.0618	7 0.0390	8 0.0232	9 0.0130	10 0.0069						
	8 0.0232	9 0.0130	10 0.0069	11 0.0035	12 0.0016	13 0.0007	14 0.0003	15 0.0001	16 0.0001	17 0.0000						
15.8999	1 0.2497	2 0.2126	3 0.1709	4 0.1295	5 0.0924	6 0.0621	7 0.0392	8 0.0234	9 0.0132	10 0.0070						
	8 0.0234	9 0.0132	10 0.0070	11 0.0035	12 0.0017	13 0.0008	14 0.0003	15 0.0001	16 0.0001	17 0.0000						
15.9999	1 0.2492	2 0.2122	3 0.1707	4 0.1295	5 0.0925	6 0.0623	7 0.0395	8 0.0236	9 0.0133	10 0.0071						
	8 0.0236	9 0.0133	10 0.0071	11 0.0036	12 0.0017	13 0.0008	14 0.0003	15 0.0001	16 0.0001	17 0.0000						
16.0999	1 0.2486	2 0.2118	3 0.1705	4 0.1295	5 0.0927	6 0.0625	7 0.0397	8 0.0238	9 0.0135	10 0.0072						
	8 0.0238	9 0.0135	10 0.0072	11 0.0036	12 0.0018	13 0.0008	14 0.0003	15 0.0001	16 0.0001	17 0.0000						
16.1999	1 0.2480	2 0.2114	3 0.1703	4 0.1295	5 0.0928	6 0.0627	7 0.0399	8 0.0240	9 0.0136	10 0.0073						
	8 0.0240	9 0.0136	10 0.0073	11 0.0037	12 0.0018	13 0.0008	14 0.0004	15 0.0001	16 0.0001	17 0.0000						
16.2999	1 0.2475	2 0.2110	3 0.1701	4 0.1295	5 0.0930	6 0.0629	7 0.0402	8 0.0242	9 0.0138	10 0.0074						
	8 0.0242	9 0.0138	10 0.0074	11 0.0038	12 0.0018	13 0.0008	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.3999	1 0.2469	2 0.2106	3 0.1699	4 0.1295	5 0.0931	6 0.0631	7 0.0404	8 0.0244	9 0.0139	10 0.0075						
	8 0.0244	9 0.0139	10 0.0075	11 0.0038	12 0.0019	13 0.0009	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.4999	1 0.2463	2 0.2102	3 0.1698	4 0.1295	5 0.0933	6 0.0634	7 0.0406	8 0.0246	9 0.0141	10 0.0076						
	8 0.0246	9 0.0141	10 0.0076	11 0.0039	12 0.0019	13 0.0009	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.5999	1 0.2458	2 0.2098	3 0.1696	4 0.1295	5 0.0934	6 0.0636	7 0.0408	8 0.0248	9 0.0142	10 0.0077						
	8 0.0248	9 0.0142	10 0.0077	11 0.0040	12 0.0019	13 0.0009	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.6998	1 0.2453	2 0.2094	3 0.1694	4 0.1295	5 0.0935	6 0.0638	7 0.0411	8 0.0250	9 0.0144	10 0.0078						
	8 0.0250	9 0.0144	10 0.0078	11 0.0040	12 0.0020	13 0.0009	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.7998	1 0.2447	2 0.2090	3 0.1692	4 0.1295	5 0.0937	6 0.0640	7 0.0413	8 0.0252	9 0.0145	10 0.0079						
	8 0.0252	9 0.0145	10 0.0079	11 0.0041	12 0.0020	13 0.0009	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.8998	1 0.2442	2 0.2086	3 0.1690	4 0.1295	5 0.0938	6 0.0642	7 0.0415	8 0.0254	9 0.0147	10 0.0080						
	8 0.0254	9 0.0147	10 0.0080	11 0.0042	12 0.0021	13 0.0010	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
16.9998	1 0.2437	2 0.2082	3 0.1688	4 0.1295	5 0.0939	6 0.0644	7 0.0417	8 0.0255	9 0.0148	10 0.0081						
	8 0.0255	9 0.0148	10 0.0081	11 0.0042	12 0.0021	13 0.0010	14 0.0004	15 0.0002	16 0.0001	17 0.0000						
17.0998	1 0.2431	2 0.2079	3 0.1686	4 0.1295	5 0.0940	6 0.0646	7 0.0419	8 0.0257	9 0.0150	10 0.0082						
	8 0.0257	9 0.0150	10 0.0083	11 0.0043	12 0.0021	13 0.0010	14 0.0005	15 0.0002	16 0.0001	17 0.0000						
17.1998	1 0.2426	2 0.2075	3 0.1684	4 0.1295	5 0.0942	6 0.0648	7 0.0421	8 0.0259	9 0.0151	10 0.0084						
	8 0.0259	9 0.0151	10 0.0084	11 0.0044	12 0.0022	13 0.0010	14 0.0005	15 0.0002	16 0.0001	17 0.0000						

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR																			
17.2998	1	0.2421	2	0.2071	3	0.1682	4	0.1294	5	0.0943	6	0.0649	7	0.0423	8	0.0261	9	0.0153	10	0.0085
	8	0.0261	9	0.0153	10	0.0085	11	0.0045	12	0.0022	13	0.0011	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.3998	1	0.2416	2	0.2067	3	0.1680	4	0.1294	5	0.0944	6	0.0651	7	0.0425	8	0.0263	9	0.0154	10	0.0086
	8	0.0263	9	0.0154	10	0.0086	11	0.0045	12	0.0023	13	0.0011	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.4998	1	0.2411	2	0.2064	3	0.1679	4	0.1294	5	0.0945	6	0.0653	7	0.0427	8	0.0265	9	0.0156	10	0.0087
	8	0.0265	9	0.0156	10	0.0087	11	0.0046	12	0.0023	13	0.0011	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.5998	1	0.2406	2	0.2060	3	0.1677	4	0.1294	5	0.0946	6	0.0655	7	0.0429	8	0.0267	9	0.0157	10	0.0088
	8	0.0267	9	0.0157	10	0.0088	11	0.0047	12	0.0024	13	0.0011	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.6998	1	0.2401	2	0.2057	3	0.1675	4	0.1294	5	0.0947	6	0.0657	7	0.0431	8	0.0269	9	0.0159	10	0.0089
	8	0.0269	9	0.0159	10	0.0089	11	0.0047	12	0.0024	13	0.0012	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.7997	1	0.2396	2	0.2053	3	0.1673	4	0.1294	5	0.0948	6	0.0659	7	0.0433	8	0.0270	9	0.0160	10	0.0090
	8	0.0270	9	0.0160	10	0.0090	11	0.0048	12	0.0024	13	0.0012	14	0.0005	15	0.0002	16	0.0001	17	0.0000
17.8997	1	0.2391	2	0.2049	3	0.1671	4	0.1293	5	0.0949	6	0.0660	7	0.0435	8	0.0272	9	0.0161	10	0.0091
	8	0.0272	9	0.0161	10	0.0091	11	0.0049	12	0.0025	13	0.0012	14	0.0006	15	0.0002	16	0.0001	17	0.0000
17.9997	1	0.2386	2	0.2046	3	0.1669	4	0.1293	5	0.0950	6	0.0662	7	0.0437	8	0.0274	9	0.0163	10	0.0092
	8	0.0274	9	0.0163	10	0.0092	11	0.0049	12	0.0025	13	0.0012	14	0.0006	15	0.0003	16	0.0001	17	0.0000
18.0997	1	0.2381	2	0.2042	3	0.1667	4	0.1293	5	0.0951	6	0.0664	7	0.0439	8	0.0276	9	0.0164	10	0.0093
	8	0.0276	9	0.0164	10	0.0093	11	0.0050	12	0.0026	13	0.0013	14	0.0006	15	0.0003	16	0.0001	17	0.0000
18.1997	1	0.2377	2	0.2039	3	0.1666	4	0.1293	5	0.0952	6	0.0666	7	0.0441	8	0.0278	9	0.0166	10	0.0094
	8	0.0278	9	0.0166	10	0.0094	11	0.0051	12	0.0026	13	0.0013	14	0.0006	15	0.0003	16	0.0001	17	0.0000
18.2997	1	0.2372	2	0.2035	3	0.1664	4	0.1292	5	0.0953	6	0.0667	7	0.0443	8	0.0279	9	0.0167	10	0.0095
	8	0.0279	9	0.0167	10	0.0095	11	0.0052	12	0.0027	13	0.0013	14	0.0006	15	0.0003	16	0.0001	17	0.0000
18.3997	1	0.2367	2	0.2032	3	0.1662	4	0.1292	5	0.0954	6	0.0669	7	0.0445	8	0.0281	9	0.0169	10	0.0096
	8	0.0281	9	0.0169	10	0.0096	11	0.0052	12	0.0027	13	0.0013	14	0.0006	15	0.0003	16	0.0001	17	0.0000
18.4997	1	0.2363	2	0.2029	3	0.1660	4	0.1292	5	0.0955	6	0.0671	7	0.0447	8	0.0283	9	0.0170	10	0.0097
	9	0.0170	10	0.0097	11	0.0053	12	0.0028	13	0.0014	14	0.0006	15	0.0003	16	0.0001	17	0.0001	18	0.0000
18.5997	1	0.2358	2	0.2025	3	0.1658	4	0.1292	5	0.0956	6	0.0672	7	0.0449	8	0.0285	9	0.0172	10	0.0098
	9	0.0172	10	0.0098	11	0.0054	12	0.0028	13	0.0014	14	0.0007	15	0.0003	16	0.0001	17	0.0001	18	0.0000
18.6997	1	0.2354	2	0.2022	3	0.1656	4	0.1291	5	0.0957	6	0.0674	7	0.0451	8	0.0286	9	0.0173	10	0.0099
	9	0.0173	10	0.0099	11	0.0054	12	0.0028	13	0.0014	14	0.0007	15	0.0003	16	0.0001	17	0.0001	18	0.0000
18.7997	1	0.2349	2	0.2019	3	0.1655	4	0.1291	5	0.0958	6	0.0675	7	0.0452	8	0.0288	9	0.0174	10	0.0101
	9	0.0174	10	0.0101	11	0.0055	12	0.0029	13	0.0014	14	0.0007	15	0.0003	16	0.0001	17	0.0001	18	0.0000
18.8996	1	0.2344	2	0.2015	3	0.1653	4	0.1291	5	0.0959	6	0.0677	7	0.0454	8	0.0290	9	0.0176	10	0.0102
	9	0.0176	10	0.0102	11	0.0056	12	0.0029	13	0.0015	14	0.0007	15	0.0003	16	0.0001	17	0.0001	18	0.0000
18.9996	1	0.2340	2	0.2012	3	0.1651	4	0.1291	5	0.0960	6	0.0678	7	0.0456	8	0.0291	9	0.0177	10	0.0103
	9	0.0177	10	0.0103	11	0.0057	12	0.0030	13	0.0015	14	0.0007	15	0.0003	16	0.0001	17	0.0001	18	0.0000
19.0996	1	0.2335	2	0.2009	3	0.1649	4	0.1290	5	0.0960	6	0.0680	7	0.0458	8	0.0293	9	0.0179	10	0.0104
	9	0.0179	10	0.0104	11	0.0057	12	0.0030	13	0.0015	14	0.0007	15	0.0003	16	0.0002	17	0.0001	18	0.0000
19.1996	1	0.2331	2	0.2005	3	0.1648	4	0.1290	5	0.0961	6	0.0681	7	0.0460	8	0.0295	9	0.0180	10	0.0105
	9	0.0180	10	0.0105	11	0.0058	12	0.0031	13	0.0016	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.2996	1	0.2327	2	0.2002	3	0.1646	4	0.1289	5	0.0962	6	0.0683	7	0.0461	8	0.0296	9	0.0181	10	0.0106
	9	0.0181	10	0.0106	11	0.0059	12	0.0031	13	0.0016	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.3996	1	0.2322	2	0.1999	3	0.1644	4	0.1289	5	0.0963	6	0.0684	7	0.0463	8	0.0298	9	0.0183	10	0.0107
	9	0.0183	10	0.0107	11	0.0060	12	0.0032	13	0.0016	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.4996	1	0.2318	2	0.1996	3	0.1642	4	0.1289	5	0.0964	6	0.0686	7	0.0465	8	0.0300	9	0.0184	10	0.0108
	9	0.0184	10	0.0108	11	0.0060	12	0.0032	13	0.0016	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.5996	1	0.2314	2	0.1993	3	0.1640	4	0.1288	5	0.0964	6	0.0687	7	0.0466	8	0.0301	9	0.0186	10	0.0109
	9	0.0186	10	0.0109	11	0.0061	12	0.0033	13	0.0017	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.6996	1	0.2310	2	0.1990	3	0.1639	4	0.1288	5	0.0965	6	0.0689	7	0.0468	8	0.0303	9	0.0187	10	0.0110
	9	0.0187	10	0.0110	11	0.0062	12	0.0033	13	0.0017	14	0.0008	15	0.0004	16	0.0002	17	0.0001	18	0.0000

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	NMAX	LAMBDA	MU	ALPHA
0	21	0.4500	0.5000	0.9000

T	P-VECTOR																	
19.7996	1	0.2305	2	0.1986	3	0.1637	4	0.1288	5	0.0966	6	0.0690	7	0.0470	8	0.0305	9	0.0188	10	0.0111
	9	0.0188	10	0.0111	11	0.0063	12	0.0034	13	0.0017	14	0.0009	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.8996	1	0.2301	2	0.1983	3	0.1635	4	0.1287	5	0.0966	6	0.0691	7	0.0471	8	0.0306	9	0.0190	10	0.0112
	9	0.0190	10	0.0112	11	0.0063	12	0.0034	13	0.0018	14	0.0009	15	0.0004	16	0.0002	17	0.0001	18	0.0000
19.9995	1	0.2297	2	0.1980	3	0.1633	4	0.1287	5	0.0967	6	0.0693	7	0.0473	8	0.0308	9	0.0191	10	0.0113
	9	0.0191	10	0.0113	11	0.0064	12	0.0035	13	0.0018	14	0.0009	15	0.0004	16	0.0002	17	0.0001	18	0.0000

APPENDIX C

RUNGE-KUTTA PROGRAM

The Chapman-Kolmogorov equations representing a queueing model are readily solvable using the Runge-Kutta method of solving differential equations. Experience has shown, however, that a technique of standardization of these equations and the form of the Runge-Kutta algorithm to solve them must be sought in order to obtain generality, convenience and efficiency. This section describes a programming package which will guide its user in obtaining a systematic time-dependent solution of the differential equations of queueing models.

It is assumed that the user of this programming package is familiar with the Chapman-Kolmogorov form of queueing equations and has a particular problem to be solved. Furthermore, it is assumed that the user is familiar with the FORTRAN-IV programming language.

The user should be acquainted enough with the Runge-Kutta algorithm to realize that it is a self-starting method which requires four evaluations of the differential equations per integration step h in the independent variable t. Preliminary evaluations are made twice with the independent variable at t+h/2, one at t+h and then one final

evaluation with correct, updated solutions at t+h. In the program, these steps are referred to by 1, 2, 3 and 4. The variable k will also be used to stand for these numbers. The term self-starting above means that only values of the solution variables and differentials at time t are needed to obtain a new solution at time t+h.

With these preliminaries in mind, it is now possible to provide a concise summary of the basic scheme which will be presented. To obtain numerical solutions of differential equations using the Runge-Kutta method requires one to specify the equations in FORTRAN notation, specify the initial conditions, specify values for constants and parameters as required by the particular problem and to specify the range of the independent variable t and the step size h. These conditions must be met by the user for any problem.

Normally, in addition to specifying conditions relating directly to his problem, the user must implement the integration scheme, define some type of data structure for intermediate or working storage and construct an output format and control over print frequency.

The programming package presented here standardizes the method of defining the necessary conditions for any problem mentioned formerly and thus eliminates the necessity to plan, program and debug the latter steps relating to implementation detail. This is accomplished by requiring

the user to construct a subroutine named INIT with FORTRAN Entry points named PRIME, NEWVAL and WRAPUP. This subroutine will describe the user problem in standard form to the programming package.

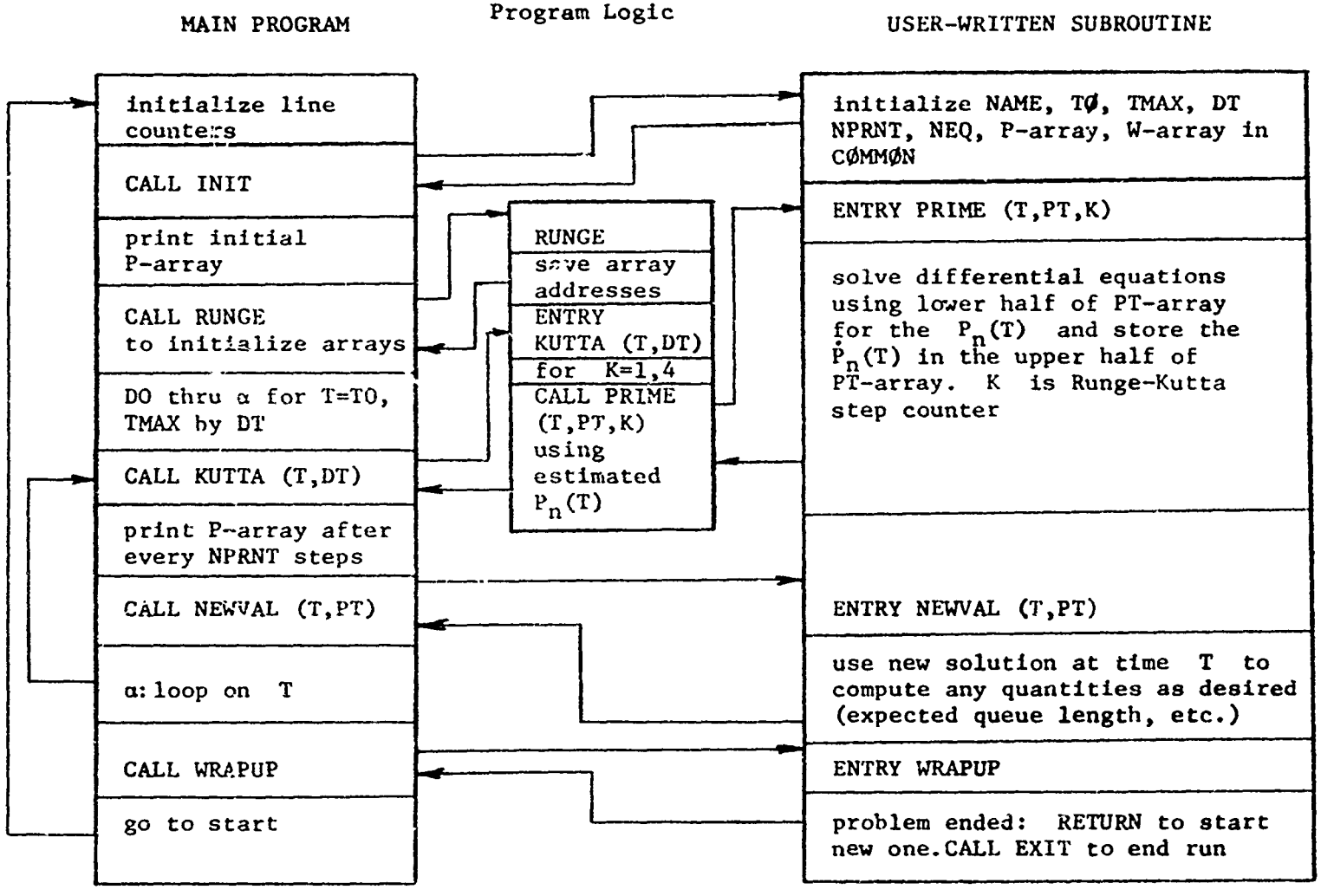
Briefly, initial entry point INIT will be used by the user to specify initial conditions of the problem, to specify the range of the independent variable t and the step size. These data will be assigned to variables listed in a specified order in a CØMMØN statement. Entry point PRIME will serve to provide computed values of the differentials, given values of the solution variable during operation of the generalized Runge-Kutta subroutine Through entry point NEWVAL, the package makes the solution variables and updated differentials available after each step h in t. Entry point WRAPUP is used after the problem is solved to allow the user to decide whether to start a new problem or terminate the computer run.

Table three Program Logic presented below shows how the user subroutine INIT interfaces with the programming package, which consists of a Main program and the Runge-kutta subroutine. It is not necessary for a user to understand the programming package itself, but only to understand the responsibilities required of the INIT subroutine at each of the points of interface.

Following the Program Logic, the next part of this section will consist of a detailed example of how the

equations for the M/M/1 queue were solved using this systematic programming package.

Table 3
Program Logic



Solution of the M/M/1 Queue

The equations for the M/M/1 queue are

$$\dot{P}_0(t) = -\lambda P_0(t) + \mu P_1(t) ,$$

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda+\mu) P_n(t) + \mu P_{n+1}(t), \text{ for } 1 \leq n \leq N,$$

$$\dot{P}_N(t) = \lambda P_{N-1}(t) - (\lambda+\mu) P_N(t)$$

where N is the maximum number of items in the system.

In this example $P_0(0)=1$ and $P_n(0)=0$ for n=1;N are the initial conditions. The parameters are $\lambda=0.45$, $\mu=0.5$ and N=20. The variable t will run from 0 to 20 with Runge-Kutta step size of 0.1.

Although theoretically there is no reason to limit the maximum number in the system, in actual computation an upper bound must be chosen. One can either truncate negligible terms or modify the equations to take care of the finite waiting room, as was done above. The latter choice is recommended because it is then possible to utilize the auxiliary condition

$$\sum_{n=0}^{\infty} P_n(t) = 1$$

to compute a measure of solution error $\varepsilon(t)$ from

$$\varepsilon(t) = 1 - \sum_{n=0}^{N} \hat{P}_n(t) \ .$$

The $\hat{P}_n(t)$ notation is used to distinguish between the notation used in the model and the actual computed values.

The usual practice is to assign initial conditions at t=0 and obtain the solution of the equations at selected positive values of t. The programming package will print the solution after every multiple of NPRNT Runge-Kutta steps chosen by the user.

Subroutine INIT

At this point, we have a statement of the necessary conditions of the sample problem and are ready to write a subroutine INIT for solving it. The following will be a step-by-step procedure for writing the FORTRAN subroutine INIT. Each FORTRAN statement will be followed by an explanation of it in relation to the problem to be solved. Statements marked by ✓ indicate the statement is always required by the programming package interface for any problem, except for the characters underscored, which are to be provided by the user as required by the problem. Statements not so indicated are user-specified to pertain to the problem at hand.

```
✓  1)      SUBRØUTINE INIT
✓  2)      CØMMØN NAME(20),TØ,TMAX,DT,NPRNT,NEQ,
```

P(21,2),W(63)

This statement is used as a common reference to locate data for the problem. Each array except W and each variable in CØMMØN must be assigned values as described below.

3) REAL LAMBDA,MU

Indicates λ and μ , respectively, in the equations will have these names and will be floating point reals.

✓ 4) READ (5,1,END=10) NAME,LAMBDA,MU,TMAX,DT

✓ 5) 1 FØRMAT (20A4/4F10.0)

Reads a problem identification card from the card reader and stores in array NAME. Reads the parameters λ and μ , the upper limit of time t for the equations TMAX and the step size for Runge-Kutta integration h, named DT in the program.

6) PRINT 2,NAME,LAMBDA,MU,TMAX,DT

7) 2 FØRMAT (1H1 20A4/1H0 9X6HLAMBDA 13X2HMU
11X4HTMAX 13X2HDT/1H0 4F15.4)

Prints out the input data.

✓ 8) TØ=0.0

Assign the initial value of time t, named T in the program. This could have also received its value in the READ statement as well. The problem is to be integrated from TØ to TMAX in steps of DT.

✓ 9) NPRNT=5

This is the number of DT steps in time t between printouts of the solution by the package Main program.

✓ 10) NEQ=21

The problem has 21 equations to be solved. Since FORTRAN does not allow a subscript of zero, all subscripts of the problem must be increased by one for the program.

11) NMØ=NEQ-1

12) NPØ=NEQ+1

13) CØNS=LAMBDA+MU

Constants used by this problem to control loops and prevent needless computation in later entry point sections of the subroutine.

✓ 14) DØ 3 I=2,NEQ

✓ 15) 3 P(I,1)=0.0

✓ 16) P(1,1)=1.0

Initialization of the P-array. A later section titled Dimensioning the Problem explains the reason for an additional subscript for the P-array.

17) LINE=55

18) IPRNT=0

Auxiliary counters used by the problem program for additional output control.

✓ 19) RETURN

Returns to the Main program to complete problem initialization.

PRIME Entry Point

This is the entry point called from the Runge-Kutta subroutine during the four step estimation of the solution vector. The estimates are in an array named PT which is given to this entry point along with time T and the Runge-Kutta step number K. The user must use the values of PT(*,1) in the differential equations which are stored in PT(*,2). The * notation here stands for the subscripting pertaining to the users' problem, explained below in the section titled Dimensioning the Problem.

```
✓ 20)        ENTRY PRIME(T,PT,K)
✓ 21)        DIMENSIØN PT(21,2)
```

The PT-array must be dimensioned exactly as the P-array was in the CØMMØN statement above.

```
✓ 22)        PT(1,2)=RHS( MU*PT(2,1)-LAMBDA*PT(1,1) )
```

First user problem equation solved and stored in upper half of P-array. Enclosing the expression in parentheses as an argument to the RHS function prevents machine underflow interrupts.

```
✓ 23)        DØ 4 I=2,NMØ
```

Loop counter for the general equation of user problem.

```
✓ 24)      4 PT(I,2)=RHS( LAMBDA*PT(I-1,1)-CØNS*PT(I,1)
                    +MU*PT(I+1,1) )
```

The general equation of the user problem is computed and stored in the upper half of the PT-array.

```
✓ 25)        PT(NEQ,2)=RHS( LAMBDA*PT(NMØ,1)-CØNS*PT(NEQ,1) )
```

The final equation for the user problem is computed and stored in the upper half of the PT-array.

✓ 26) RETURN

Return control to the Runge-Kutta subroutine.

NEWVAL Entry Point

This entry point is called after each step in the solution, i.e., after each time that a new solution vector is computed and T←T+DT. The user is given the new value of time T and the solution vector and updated derivatives in the PT-array.

✓ 27) ENTRY NEWVAL(T,PT)

In this problem it is desired to compute and print the solution error after every NPRNT steps of DT in T and label the output at the top of each page.

28) CALL CØNTRL(IPRNT,NPRNT,&11)

CØNTRL is part of the programming package. Using the free counter variable IPRNT and counter limit NPRNT, it returns to statement 11 whenever IPRNT is less then NPRNT after incrementing IPRNT by one. It returns to the next statement if IPRNT equals NPRNT after setting IPRNT to zero.

29) CALL CØNTRL(LINE,55,&5)

This counter switch, used for page control, is similar to the above. It executes the next statement after every 55 passes.

30) PRINT 7,NAME

Print column headings at top of each page, that is, after every 55 passes through this section.

```
31)     7  FØRMAT (1H1 20A4/1H0 14X 1HT 6X
                14HERRØR, PERCENT)
32)     5  SUM=0.0
```

Initialize for computing the solution error.

```
33)        DØ 8 I=1,NEQ
```

Loop for computing sum of solution vector.

```
34)     8  SUM=SUM+PT(NPØ-I,1)
35)        ERRØR=100.0*(SUM-1.0)
```

Compute percent error of solution.

```
36)        PRINT 9,T,ERRØR
```

Print current time T and percent error.

```
37)     9  FØRMAT(1H F15.4,F20.6)
✓ 38)  11  RETURN
```

Any problem statistics such as the expected number in the system would be computed in entry point NEWVAL by the user as required by his particular problem.

WRAPUP Entry Point

This entry point is called from the Main program after the problem has been solved, that is, when time T reaches TMAX. If the user executes a RETURN, the Main program calls INIT, thus providing the user the opportunity to initialize a new problem. A CALL EXIT statement, written instead, terminates the run.

```
✓ 39)        ENTRY WRAPUP
  40)        RETURN
```

In this example, the RETURN causes a call to INIT. The READ statement is executed to get new problem parameters. If no more data is in the reader, a branch is made to statement 10 to terminate the run.

```
  41)    10  CALL EXIT
✓ 42)        END
```

The FORTRAN listing of this program and sample output are included next. The section following the listings will discuss in detail the problem of storage of the probability vectors in the P-array.

```
0001                  SUBROUTINE INIT
                C
                C***  SIMPLE M/M/1 QUEUE.
                C
0002                  COMMON NAME(20),T0,TMAX,DT,NPRNT,NEQ,P(21,2),W(63)
                C
                C***  THE SYSTEM HAS 21 STATES AND IS OF ORDER 1, THEREFORE THE P-MATRIX
                C***  IS DIMENSIONED 21 BY 2. THE W ARRAY IS ALWAYS 1.5 TIMES THE SIZE
                C***  OF P, THEREFORE IT IS DIMENSIONED FOR 21*2*1.5=63.
                C
0003                  REAL LAMBDA,MU
0004                  READ (5,1,END=10) NAME,LAMBDA,MU,TMAX,DT
0005                1 FORMAT (20A4/4F10.0)
0006                  PRINT 2,NAME,LAMBDA,MU,TMAX,DT
0007                2 FORMAT (1H1,20A4//1HO 9X6HLAMBDA 13X2HMU 11X4HTMAX 13X2HDT/
                     . 1H0,4F15.4)
0008                  T0=0.0
0009                  NPRNT=5
0010                  NEQ=21
0011                  NM1=NEQ-1
0012                  NP1=NEQ+1
0013                  CONS=LAMBDA+MU
0014                  DO 3 I=2,NEQ
0015                3 P(I,1)=0.0
0016                  P(1,1)=1.0
0017                  LINE=55
0018                  IPAGE=0
0019                  RETURN
                C
0020                  ENTRY PRIME(T,PT,K)
0021                  DIMENSION PT(21,2)
                C
                C***  COMPUTE THE DIFFERENTIAL-DIFFERENCE EQUATIONS.
                C***  SINCE THIS IS WITHIN THE INTEGRATION LOOP, THE PT MATRIX IS
                C***  FROM WORKING STORAGE. THIS IS WHY A DIFFERENT NAME MUST BE USED.
                C
0022                  PT(1,2)=RHS(MU*PT(2,1)-LAMBDA*PT(1,1))
0023                  DO 4 I=2,NM1
0024                4 PT(I,2)=RHS(LAMBDA*PT(I-1,1)-CONS*PT(I,1)+MU*PT(I+1,1))
0025                  PT(NEQ,2)=RHS(LAMBDA*PT(NM1,1)-CONS*PT(NEQ,1))
0026                  RETURN
                C
0027                  ENTRY NEWVAL(T,PT)
                C
                C***  IN THIS ENTRY SECTION THE USER IS GIVEN THE UPDATED SOLUTION IN
                C***  THE PT ARRAY ALONG WITH UPDATED TIME T. THE USER IS FREE TO
                C***  COMPUTE ANY STATISTICS NEEDED SUCH AS THE EXPECTED NUMBER IN THE
                C***  SYSTEM. IN THIS PROBLEM, WE WILL ONLY COMPUTE THE SOLUTION ERROR.
```

```
      C
      C***  COMPUTE THE ERROR AND PRINT AFTER EVERY NPRNT STEPS.
      C
0028            CALL CONTRL(IPRNT,NPRNT,611)
0029            CALL CONTRL(LINE,55,65)
0030            PRINT 7,NAME
0031          7 FORMAT (1H1,2A4,1HO 14XIHT 6X14HERROR, PERCENT)
0032          5 SUM=0.0
0033            DO 8 I=1,NEQ
0034          8 SUM=SUM+PT(NPO-I,1)
0035            ERROR=100.0*(SUM-1.0)
0036            PRINT 9,T,ERROR
0037          9 FORMAT (1H F15.4,F20.6)
0038         11 RETURN
      C
0039            ENTRY WRAPUP
0040            RETURN
0041         10 CALL EXIT
0042            END
```

```
C***  MAIN CONTROL MODULE FOR RUN -KUTTA QUEUEING EQUATION SOLVER.
C***  STUART OLSON -- APRIL 1972 -UNIV. OF IOWA -- 56-299
C
0001        COMMON NAME(20),TO,TMAX,DT,NPRNT,NEQ,P(1)
0002        DATA EPS/5.0E-5/
0003        INTEGER *2 IP(2)/' ','0'/,IH
0004     13 CALL INIT
0005        NSTAR=1
0006        DO 2 I=1,NEQ
0007          IF (P(I).GT.0.0) NSTAR=I
0008      2 CONTINUE
0009        NP0=NEQ+1
0010        T=TO
C
C***  SOLVE DIFF-DIFF EQNS. AT INITIAL CONDITIONS.
C
0011        CALL PRIME(T,P,1)
C
C***  MAKE WORKING-STORAGE ASSIGNMENTS TO RUNGE-KUTTA SUBROUTINE.
C
0012        CALL RUNGE(P(1),P(2*NEQ+1),P(4*NEQ+1),NEQ,1,2)
C
C***  PRINT-OUT INITIAL P-VECTOR.
C
0013        WRITE (8,3) NAME
0014      3 FORMAT (1H1,2(A4,1H0 8X),1HT,3X,8HP-VECTOR)
0015        N=MIN0(10,NSTAR)
0016        IH=IP(2)
0017        IF (N.LE.10) IH=IP(1)
0018        WRITE (8,4) IH,T,(I,P(I),I=1,N)
0019      4 FORMAT (A1,F9.2,10(I4,F7.4))
0020        IF (NSTAR.LE.10) GO TO 5
0021        N=N+1
0022        WRITE (8,6) (I,P(I),I=N,NSTAR)
0023      6 FORMAT (1H ,9X,I4,F7.4,I4,F7.4,I4,F7.4,I4,F7.4,I4,F7.4,I4,F7.4,
     .  I4,F7.4,I4,F7.4,I4,F7.4,I4,F7.4)
0024      5 LINE=-52+(NSTAR+9)/10
0025        IPRNT=0
0026      7 NT=(TMAX-TO)/DT+0.5
C
C***  START INTEGRATION LOOP. STEP FROM TO TO TMAX IN STEPS OF DT.
C
0027        DO 8 IT=1,NT
C
C***  SOLVE DIFF-DIFF EQNS. FOR T=T+DT.
C
0028        CALL KUTTA(T,DT)
C
```

```
C***  TRANSFER TO USER-WRITTEN ENTRY POINT WITH NEW SOLUTION VECTOR.
C
0029        CALL NEWVAL(T,P)
C
C***  PRINT-OUT P-VECTOR AFTER EVERY NPRNT STEPS.
C
0030          IF (IT.EQ.NT) GO TO 14
0031          CALL CONTRL(IPRNT,NPRNT,68)
0032       14 DO 9 I=1,NPO
0033            NSTAR=NPO-I
0034            IF (P(NSTAR).GE.EPS) GO TO 10
0035        9   CONTINUE
0036          NSTAR=NPO
0037       10 LINE=LINE+(NSTAR+9)/10
0038          IF (LINE)11,12,12
0039       12 LINE=54+(NSTAR+9)/10
0040          WRITE (6,3) NAME
0041       11 N=MIN0(10,NSTAR)
0042          IN=IP(2)
0043          IF (N.LE.10) IX=IP(1)
0044          WRITE (6,4) IN,T,(I,P(I),I=1,N)
0045            IF (NSTAR.LE.10) GO TO 8
0046            N=N+1
0047          WRITE (6,6) (I,P(I),I=N,NSTAR)
0048        8 CONTINUE
C
C***  CALL USER-WRITTEN ENTRY POINT FOR PROBLEM-TERMINATION DECISION.
C
0049        CALL WRAPUP
C
C***  RETURN TO INITIAL ENTRY POINT IF CONTROL HAS BEEN RETURNED.
C
0050        GO TO 13
0051        END
```

```
0001            SUBROUTINE RUNGE(V,W,D,NEQ,NORD,NORDP1)
0002            DIMENSION V(NEQ,NORDP1),W(NEQ,NORDP1),D(NEQ,NORD)
                C
                C***  GENERALIZED RUNGE-KUTTA SUBROUTINE. ENTRY POINT RUNGE IS USED TO
                C***  MAKE WORKING-STORAGE ASSIGNMENTS. V IS THE PERMANENT STORAGE
                C***  MATRIX OF SOLUTION EQUATIONS AND THE VALUES OF THE DIFFERENTIALS
                C***  AT TIME T. W IS A WORKING STORAGE MATRIX OF THE SAME SIZE.
                C***  D IS ANOTHER WORKING STORAGE MATRIX WITH ONE LESS COLUMN AS W.
                C***  NEQ IS THE NUMBER OF EQUATIONS TO BE SOLVED.
                C***  NORD IS THE ORDER OF THE EQUATIONS TO BE SOLVED.
                C***  NORDP1 IS JUST NORD+1.
                C
0003            RETURN
                C
                C***  ENTRY POINT KUTTA IS USED TO ADVANCE THE SOLUTION FROM T TO T+DT.
                C***  IT CALLS THE USER-WRITTEN ENTRY POINT PRIME FOUR TIMES FOR
                C***  EVALUATION OF THE DIFFERENTIAL EQNS. FOR DIFFERENT ESTIMATES OF
                C***  THE SOLUTION VARIABLES. WHEN CONTROL IS RETURNED, THE NEW SOLUTION
                C***  VARIABLES AND CORRESPONDING VALUES OF DERIVATIVES ARE IN THE V
                C***  MATRIX.
                C
0004            ENTRY KUTTA(T,DT)
0005            DT2=DT/2.0
0006            DT6=DT/6.0
0007            DO 1 J=2,NORDP1
0008            JM1=J-1
0009            DO 1 I=1,NEQ
0010            W(I,JM1)=V(I,JM1)
0011          1 W(I,J)=V(I,J)
0012            DO 6 I=1,NEQ
0013          6 W(I,NORDP1)=V(I,NORDP1)
0014            DO 2 K=1,3
0015            IF(K-2)3,3,4
0016          3 DT=DT2
0017            W=0.5
0018            GO TO 4
0019          3 DT=DT
0020            W=1.0
0021          4 TW=T+DTW
0022            DO 5 J=2,NORDP1
0023            JM1=J-1
0024            DO 5 I=1,NEQ
0025          5 W(I,JM1)=V(I,JM1)+W(I,J)*DTW
0026            CALL PRIME(TW,W,K)
0027            DO 2 J=2,NORDP1
0028            JM1=J-1
0029            DO 2 I=1,NEQ
0030          2 D(I,JM1)=D(I,JM1)+W(I,J)*WT
```

```
0031              DO 7 J=1,NGRD
0032              DO 7 I=1,NEQ
0033            7 V(I,J)=V(I,J)+DT6*D(I,J)
0034              T=TW
0035              CALL PRIME(T,V,4)
0036              RETURN
0037              END
```

```
FORTRAN IV G LEVEL  20                RHS              DATE = 72103        18/11/36          PAGE 0001

0001                  FUNCTION RHS(P)
                      C
                      C***  THIS FUNCTION IS REQUIRED TO PREVENT EXPONENT UNDERFLOW BY
                      C***  ARBITRARILY CUTTING OFF SMALL ABSOLUTE VALUES BEFORE THEY
                      C***  CAN CAUSE A MACHINE UNDERFLOW. (ON 360/65 UNDERFLOW OCCURS WHEN
                      C***  A NUMBER IS LESS THAN 16**64 AFTER A COMPUTATION.)
                      C
0002                  REAL EPS/1.0E-70/
0003                  IF (ABS(P).LT.EPS) GO TO 1
0004                  RHS=P
0005                  RETURN
0006                1 RHS=0.0
0007                  RETURN
0008                  END
```

```
FORTRAN IV G LEVEL  20                  CONTRL              DATE = 72103          18/11/36              PAGE 0001

0001                  SUBROUTINE CONTRL(I,N,*)
                      C
                      C***  THIS SUBROUTINE HAS TWO EXIT POINTS -- ONE FOR WHEN I<N AND
                      C***  ONE FOR I=N, IN WHICH CASE I IS ALSO SET TO ZERO.
                      C
0002                  I=I+1
0003                  IF (I.LT.N) RETURN 1
0004                  I=0
0005                  RETURN
0006                  END
```

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S.OLSON

LAMBDA	MU	TMAX	DT
0.4500	0.5000	20.0000	0.1000

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S.OLSON

T	ERROR, PERCENT
0.5000	-0.000012
1.0000	-0.000036
1.5000	-0.000077
2.0000	-0.000125
2.5000	-0.000167
3.0000	-0.000221
3.5000	-0.000268
4.0000	-0.000334
4.5000	-0.000387
5.0000	-0.000435
5.5000	-0.000501
6.0000	-0.000548
6.5000	-0.000608
7.0000	-0.000668
7.5000	-0.000721
8.0000	-0.000781
8.5000	-0.000846
9.0000	-0.000912
9.5000	-0.000989
9.9999	-0.001061
10.4999	-0.001132
10.9999	-0.001192
11.4999	-0.001276
11.9999	-0.001335
12.4999	-0.001413
12.9999	-0.001502
13.4999	-0.001597
13.9999	-0.001675
14.4999	-0.001734
14.9999	-0.001895
15.4999	-0.002027
15.9999	-0.002170
16.4999	-0.002349
16.9998	-0.002551
17.4998	-0.002913
17.9997	-0.003117
18.4997	-0.003475
18.9996	-0.003910
19.4996	-0.004423
19.9995	-0.005013

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S.OLSON

T	P-VECTOR											
0.0	1 1.0000											
0.50	1 0.8194	2 0.1619	3 0.0174	4 0.0013	5 0.0001							
1.00	1 0.7009	2 0.2421	3 0.0492	4 0.0070	5 0.0008	6 0.0001						
1.50	1 0.6189	2 0.2810	3 0.0805	4 0.0166	5 0.0026	6 0.0003						
2.00	1 0.5594	2 0.2985	3 0.1068	4 0.0282	5 0.0058	6 0.0010	7 0.0001					
2.50	1 0.5144	2 0.3049	3 0.1276	4 0.0404	5 0.0102	6 0.0021	7 0.0004	8 0.0001				
3.00	1 0.4792	2 0.3054	3 0.1433	4 0.0521	5 0.0153	6 0.0037	7 0.0008	8 0.0001				
3.50	1 0.4507	2 0.3028	3 0.1552	4 0.0629	5 0.0209	6 0.0058	7 0.0014	8 0.0003	9 0.0001			
4.00	1 0.4272	2 0.2985	3 0.1639	4 0.0726	5 0.0266	6 0.0083	7 0.0022	8 0.0005	9 0.0001			
4.50	1 0.4074	2 0.2936	3 0.1703	4 0.0810	5 0.0323	6 0.0110	7 0.0033	8 0.0009	9 0.0002			
5.00	1 0.3904	2 0.2883	3 0.1749	4 0.0884	5 0.0378	6 0.0140	7 0.0045	8 0.0013	9 0.0003	10 0.0001		
5.50	1 0.3757	2 0.2829	3 0.1782	4 0.0947	5 0.0431	6 0.0170	7 0.0059	8 0.0018	9 0.0005	10 0.0001		
6.00	1 0.3627	2 0.2777	3 0.1804	4 0.1002	5 0.0480	6 0.0201	7 0.0074	8 0.0025	9 0.0007	10 0.0002		
6.50	1 0.3512	2 0.2726	3 0.1819	4 0.1048	5 0.0526	6 0.0232	7 0.0091	8 0.0032	9 0.0010	10 0.0003		
	11 0.0001											
7.00	1 0.3409	2 0.2677	3 0.1827	4 0.1088	5 0.0568	6 0.0263	7 0.0108	8 0.0040	9 0.0014	10 0.0004		
	11 0.0001											
7.50	1 0.3316	2 0.2630	3 0.1832	4 0.1122	5 0.0607	6 0.0292	7 0.0126	8 0.0049	9 0.0017	10 0.0006		
	11 0.0002											
8.00	1 0.3231	2 0.2586	3 0.1832	4 0.1150	5 0.0643	6 0.0321	7 0.0145	8 0.0059	9 0.0022	10 0.0007		
	11 0.0002	12 0.0001										
8.50	1 0.3154	2 0.2544	3 0.1830	4 0.1175	5 0.0675	6 0.0349	7 0.0163	8 0.0069	9 0.0027	10 0.0010		
	11 0.0003	12 0.0001										
9.00	1 0.3084	2 0.2504	3 0.1826	4 0.1195	5 0.0704	6 0.0375	7 0.0181	8 0.0080	9 0.0032	10 0.0012		
	11 0.0004	12 0.0001										
9.50	1 0.3018	2 0.2466	3 0.1820	4 0.1213	5 0.0731	6 0.0400	7 0.0199	8 0.0091	9 0.0038	10 0.0015		
	11 0.0005	12 0.0002	13 0.0001									
10.00	1 0.2958	2 0.2430	3 0.1813	4 0.1228	5 0.0756	6 0.0424	7 0.0217	8 0.0102	9 0.0044	10 0.0018		
	11 0.0007	12 0.0002	13 0.0001									
10.50	1 0.2902	2 0.2395	3 0.1805	4 0.1240	5 0.0778	6 0.0446	7 0.0235	8 0.0114	9 0.0051	10 0.0021		
	11 0.0008	12 0.0003	13 0.0001									
11.00	1 0.2850	2 0.2363	3 0.1796	4 0.1251	5 0.0798	6 0.0467	7 0.0252	8 0.0125	9 0.0058	10 0.0025		
	11 0.0010	12 0.0004	13 0.0001									
11.50	1 0.2801	2 0.2331	3 0.1787	4 0.1260	5 0.0816	6 0.0487	7 0.0268	8 0.0137	9 0.0065	10 0.0029		
	11 0.0012	12 0.0005	13 0.0002	14 0.0001								
12.00	1 0.2755	2 0.2302	3 0.1778	4 0.1267	5 0.0833	6 0.0506	7 0.0285	8 0.0148	9 0.0072	10 0.0033		
	11 0.0014	12 0.0006	13 0.0002	14 0.0001								
12.50	1 0.2712	2 0.2273	3 0.1768	4 0.1273	5 0.0848	6 0.0524	7 0.0300	8 0.0160	9 0.0079	10 0.0037		
	11 0.0016	12 0.0007	13 0.0003	14 0.0001								
13.00	1 0.2671	2 0.2246	3 0.1758	4 0.1277	5 0.0862	6 0.0540	7 0.0315	8 0.0171	9 0.0087	10 0.0041		
	11 0.0018	12 0.0008	13 0.0003	14 0.0001								
13.50	1 0.2633	2 0.2220	3 0.1747	4 0.1281	5 0.0874	6 0.0556	7 0.0330	8 0.0182	9 0.0095	10 0.0046		
	11 0.0021	12 0.0009	13 0.0004	14 0.0001	15 0.0001							
14.00	1 0.2596	2 0.2195	3 0.1737	4 0.1284	5 0.0886	6 0.0570	7 0.0343	8 0.0193	9 0.0102	10 0.0051		
	11 0.0024	12 0.0010	13 0.0004	14 0.0002	15 0.0001							
14.50	1 0.2562	2 0.2172	3 0.1727	4 0.1286	5 0.0896	6 0.0584	7 0.0357	8 0.0204	9 0.0110	10 0.0056		
	11 0.0027	12 0.0012	13 0.0005	14 0.0002	15 0.0001							
15.00	1 0.2529	2 0.2149	3 0.1717	4 0.1287	5 0.0905	6 0.0597	7 0.0370	8 0.0215	9 0.0118	10 0.0061		
	11 0.0030	12 0.0014	13 0.0006	14 0.0002	15 0.0001							
15.50	1 0.2498	2 0.2127	3 0.1707	4 0.1288	5 0.0914	6 0.0609	7 0.0382	8 0.0225	9 0.0125	10 0.0066		
	11 0.0033	12 0.0015	13 0.0007	14 0.0003	15 0.0001							

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIE.CY OF A-K METHOD S.OLSON

T	P-VECTOR										
16.00	1 0.2468	2 0.2156	3 0.1697	4 0.1288	5 0.0921	6 0.0620	7 0.0394	8 0.0235	9 0.0133	10 0.0071	
	11 0.0036	12 0.0017	13 0.0008	14 0.0003	15 0.0001	16 0.0001					
16.50	1 0.2446	2 0.2185	3 0.1687	4 0.1289	5 0.0928	6 0.0631	7 0.0405	8 0.0245	9 0.0140	10 0.0076	
	11 0.0039	12 0.0019	13 0.0009	14 0.0004	15 0.0002	16 0.0001					
17.00	1 0.2413	2 0.2066	3 0.1677	4 0.1288	5 0.0935	6 0.0641	7 0.0416	8 0.0255	9 0.0148	10 0.0081	
	11 0.0042	12 0.0021	13 0.0010	14 0.0004	15 0.0002	16 0.0001					
17.50	1 0.2387	2 0.2047	3 0.1667	4 0.1287	5 0.0940	6 0.0650	7 0.0426	8 0.0264	9 0.0155	10 0.0087	
	11 0.0046	12 0.0023	13 0.0011	14 0.0005	15 0.0002	16 0.0001					
18.00	1 0.2362	2 0.2029	3 0.1658	4 0.1286	5 0.0946	6 0.0659	7 0.0436	8 0.0273	9 0.0162	10 0.0092	
	11 0.0049	12 0.0025	13 0.0012	14 0.0006	15 0.0003	16 0.0001					
18.50	1 0.2339	2 0.2011	3 0.1648	4 0.1284	5 0.0950	6 0.0668	7 0.0445	8 0.0282	9 0.0170	10 0.0097	
	11 0.0053	12 0.0027	13 0.0014	14 0.0006	15 0.0003	16 0.0001	17 0.0001				
19.00	1 0.2316	2 0.1995	3 0.1639	4 0.1283	5 0.0955	6 0.0675	7 0.0454	8 0.0290	9 0.0177	10 0.0102	
	11 0.0057	12 0.0030	13 0.0015	14 0.0007	15 0.0003	16 0.0001	17 0.0001				
19.50	1 0.2294	2 0.1978	3 0.1630	4 0.1281	5 0.0958	6 0.0683	7 0.0463	8 0.0299	9 0.0184	10 0.0108	
	11 0.0060	12 0.0032	13 0.0016	14 0.0008	15 0.0004	16 0.0002	17 0.0001				
20.00	1 0.2273	2 0.1962	3 0.1621	4 0.1279	5 0.0962	6 0.0690	7 0.0471	8 0.0307	9 0.0190	10 0.0113	
	11 0.0064	12 0.0035	13 0.0018	14 0.0009	15 0.0004	16 0.0002	17 0.0001				

Dimensioning the Problem

The P-array in CØMMØN storage contains the solution vector and the derivatives. The dimension of the array, denoted by * below, is constructed to correspond to the possibly several subscripts of the user problem according to normal FORTRAN conventions. For each probability there is a corresponding derivative with the same subscript. By appending an additional subscript on the P-array with the value of two, two identical problem areas are set up. The first, that is, the one with the appended subscript value of one, is for the solution vector probabilities. The second is for the corresponding derivatives. The compiler assigns the storage in ascending order so that the first half is the lower part and the second is the upper part. Thus P(*,2) in the CØMMØN statement sets up two sequential arrays each of dimension *, where * here stands for the dimension of the user problem as discussed below in more detail. The user must assign NEQ a value equal to the product of the sizes of each component of * above. The main program is then able to locate the derivative locations by simply adding NEQ to the value of the subscript of the solution vector expression during the computations.

The Runge-Kutta subroutine requires working storage for intermediate results. This depends on the size of the user problem and thus must be assigned by the user in the CØMMØN statement as the W-array. The W-array will always be

dimensioned as exactly one-and-one-half times the size of the P(*,2)-array. An example will be given to help clarify this point.

Suppose a problem has equations

$$\dot{P}_{ijk}(t) = f(t,P_{ijk}(t))$$

where i,j and k are indicies used to enumerate the state of the system. Suppose further for this problem that

$$1 \leq i \leq 21$$
$$1 \leq j \leq 11$$
$$1 \leq k \leq 6 .$$

In the notation used above, the problem dimension * is (21,11,6). The P-array, however, must be dimensioned for this problem with an appended subscript of value two. Therefore, in order to make space available for corresponding derivative values, write P(21,11,6,2). Now the W-array must be one-and-one-half the size of the P-array. Write W(4158), which is the product (21)(11)(6)(2)(1.5).

Occasionally, a problem may be written which has more subscripts than the compiler will allow. (For the IBM 360 FORTRAN-IV compiler the maximum is seven.) Suppose the problem appears as

$$P_{i_1 i_2 i_3 \cdots i_p}(t) .$$

There are p subscripts. Also suppose these run from one to $m_1, m_2, m_3, \ldots, m_p$, respectively. The p subscripts must be replaced by a lower number of subscripts for the compiler. Say one is used, denoted by j. Given a set of p indicies in the program, the following algorithm should be used to compute j which will be then used as the subscript of the P-array.

Initially set $j=i_p-1$. Then compute

$$j \leftarrow j m_{p-k} + i_{p-k} - 1 \quad \text{for} \quad k=1 \text{ to } p-1,$$

and then compute

$$j \leftarrow j+1.$$

To illustrate, suppose the four subscripts of the former example could not be dimensioned as P(21,11,6,2) as was done because of compiler restrictions. To use equivalent single-subscripting first dimension the P-array for the product of the subscripts, P(2772). Here $m_1=21$, $m_2=11$, $m_3=6$, $m_4=2$ and $p=4$. To locate the correct subscript for the P-array for some particular choice of subscripts, say $(i_1,i_2,i_3,i_4)=(2,4,5,1)$, use the above algorithm. Set $j=i_4-1=1-1=0$, then for k=1 to 3;

$j \leftarrow jm_3 + i_3 - 1 = (0)(6) + 5 - 1 = 4,$

$j \leftarrow jm_2 + i_2 - 1 = (4)(11) + 4 - 1 = 47,$

$j \leftarrow jm_1 + i_1 - 1 = (47)(21) + 2 - 1 = 988,$

and finally

$j \leftarrow j+1 = 988+1 = 989.$

Thus P(989) corresponds to P(2,4,5,1).

Program Output

The programming package treats the P-array as if it were singly-subscripted. During a problem solution, the P-array will be printed after every NPRNT integration steps on FORTRAN unit 8. The output will consist of the problem NAME at the top of the page with the left column labeled for time T. After each value of T printed, the P-array will be printed immediately to the right. Values of the array which are less than $5*10^{-5}$ will not be printed if they follow larger values. If the single-subscript output form of the P-array is not desired by the user, define FORTRAN data set 8 as DUMMY in the operand field of the DD control card for FT08F001 as shown below in the Job Control Cards section and

Job Control Cards

The job control language (JCL) statements for running on the IBM 360/65 computer at the University of Iowa are now presented.

```
//QUEUEING JØB (XXXXXXXX),'STU QUEUES'
// EXEC PDQFØR,PARM='LIST/PRINT,MAP'
//SØURCE DD *
     (Place source deck of programming package and subroutine
      INIT here.)
/*
//DATA DD *
     (Place any data to be read by subroutine INIT from
      unit 5 here.)
/*
//FT08F001 DD SYSØUT=A,DCB=RECFM=UA
     (To get P-array output from Main program.)
//FT08F001 DD DUMMY
     (To ignore output from Main program.)
//SYSABEND DD SYSØUT=A
     (Include if a dump is desired if abnormal termination
      such as 0C4 or 0C5 occurs.)
```